BENÍCIO JOSÉ DE SOUZA

"SOFTWARE DE UM MINICOMPUTADOR: SISTEMA BÁSICO DE CONTROLE"

"Dissertação de Mestrado" apresentada à Escola Politécnica da Universidade de São Paulo, para a obtenção do título de Mestre em Engenharia".

Área de Concentração - Engenharia de Eletricidade.

ORIENTADOR: Prof. Dr. Tamio Shimizu

São Paulo
-1976-

BENÍCIO JOSÉ DE SOUZA

"SOFTWARE DE UM MINICOMPUTADOR: SISTEMA BÁSICO DE CONTROLE"

"Dissertação de Mestrado" apresentada à Escola Politécnica da Universidade de São Paulo, para a obtenção do título de Mestre em Engenharia".

Área de Concentração - Engenharia de Eletricidade.

ORIENTADOR: Prof. Dr. Tamio Shimizu

São Paulo
-1976-

À minha esposa.

A G R A D E C I M E N T O S

Ao Prof. Dr. Tamio Shimizu pela orientação na realização deste trabalho;

Ao Prof. Dr. Antonio Marcos de Aguirra Massola pelo interesse, apoio e decisiva participação;

Aos Engenheiros João José Neto, Ting Kong Sen e Selma Shin Shimizu Melnikoff pelo incentivo e valiosas sugestões apresentadas;

Ao Prof. Dr. Antonio Hélio Guerra Vieira e à "Fundação para o Desenvolvimento Tecnológico da Engenharia" que me colocaram à disposição todos os recursos necessários para a confecção deste trabalho;

À Sra. Judite Peres de Souza, minha esposa, pela dedicação e colaboração na preparação dos textos;

À Srta. Sonia Regina Izarelli pelos trabalhos de datilografia;

Ao Humberto N.Servilha pelos serviços de impressão.

A todos os engenheiros e estagiários da F.D.T.E. e do Laboratório de Sistemas Digitais que direta ou indiretamente contribuiram para a preparação e testes dos programas relacionados com este trabalho.

R E S U M O

Este trabalho descreve a implementação de um Núcleo de programação de um Sistema Básico de Controle para o Minicomputador "PATINHO FEIO" desenvolvido no "Laboratório de Sistemas Digitais do Departamento de Eletricidade da Escola Politécnica da Universidade de São Paulo".

São consideradas inicialmente as características mais importantes dos principais tipos de Sistemas Operacionais disponíveis em minicomputadores. São apresentadas algumas técnicas utilizadas na implementação de um Sistema Básico de Controle para o minicomputador G-10, desenvolvido na "Fundação para o Desenvolvimento Tecnológico da Engenharia".

Alguns conceitos relacionados com o desenvolvimento de Sistemas Operacionais em geral são comentados no capítulo 1.

O Capítulo 2 contém a descrição de um conjunto de recursos de programação denominado "NÚCLEO" proposto como ferramenta inicial para a implementação de sistemas operacionais no "PATINHO FEIO".

Em seguida é apresentada a descrição de um Sistema Básico de Controle baseado neste "NÚCLEO".

O apêndice 1 contém um resumo das principais subrotinas utilizadas na programação da "NUCLEO".

No apêndice 2 é mostrado um exemplo de utilização de Sistema Básico de Controle.

A B S T R A C T

This work describes the implementation of a programming Nucleus and a Basic Control System for the minicomputer "PATINHO FEIO" developed in the "Laboratorio de Sistemas Digitais do Departamento de Eletricidade da Escola Politécnica da Universidade de São Paulo".

First, some important characteristics of the principal types of operating systems available for minicomputers are considered. Some techiniques used in the implementation of a basic Control System for the minicomputer G-10 developed in "Fundação para o Desenvolvimento Tecnológico da Engenharia" are presented.

Some general concepts related with the development of operating systems are commented in chapter 1.

Chapter 2 contains the description of a set of programming resources called "Nucleus".The Nucleus is proposed as a tool for the implementation of operating systems in the "PATINHO FEIO".

Then a description of a Basic Control System based in this "Nucleus" is presented.

Appendix 1 contains a summary of the principals subroutines used in programming the "Nucleus".

In appendix 2 an example of the Basic Control System operation is shouwn.

Í N D I C E

PAG.

PAG.

CAPITULO I - INTRODUÇÃO

INTRODUÇÃO

O objetivo deste trabalho é descrever a implementaçao de um Sistema Básico de controle e de um Núcleo de Programaçao destinados ao estudo e desenvolvimento de Sistemas Operacionais baseados no minicomputador "PATINHO FEIO".

O minicomputador "PATINHO FEIO" desenvolvido no LABORATÓRIO DE SISTEMAS DIGITAIS DO DEPARTAMENTO DE ELETRICIDADE DA ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO, descrito na referencia(1), tem sido até entao utilizado principalmente com o propósito de promover o treinamento de pessoal estagiário, tanto na área de "HARDWARE", através de modificações no projeto original ou inclusao de novos recursos, quanto na área de engenharia de "SOFTWARE" através da implementaçao de vários programas, notadamente na categoria de "SOFTWARE" de desenvolvimento(2).

Essas características justificaram a escolha do "PATINHO FEIO" como ferramenta na aplicação de vários conceitos atualmente utilizados na implementaçao de sistemas operacionais, a despeito de uma possível deficiência de alguns recursos de "HARDWARE" deste minicomputador. Entretanto, as soluçoes que possam ser dadas no sentido de eliminar essas deficiências foram consideradas como parte deste estudo, podendo proporcionar subsídios para outros projetos nesta área.

A definiçao dada ao NÚCLEO descrito no capítulo 2 procura abordar os principais aspectos envolvidos no estabelecimento dos recursos mínimos de programaçao que possibilitam o desenvolvimento de um sistema operacional, dentro das possibilidades de um minicomputador do porte do "PATINHO FEIO".

O Sistema Básico de Controle cuja descriçao é dada no capítulo 3, tem principalmente duas finalidades, quais sejam, a de ilustrar a aplicação dos recursos proporcionados pelo Núcleo, no desenvolvimento de um sistema operacional simples como é o Sistema Básico de Controle, e ainda, por meio deste, avaliar o desempenho do Núcleo proposto inicalmente. Esta avaliaçao deverá

orientar as possíveis modificações, otimizações ou introdução de novos recursos, tanto de "SOFTWARE" como de "HARDWARE" que viabilizem a construção de sistemas operacionais mais sofisticados que levem em conta um outro conjunto de aplicações.

1.1 - SISTEMAS OPERACIONAIS PARA MINICOMPUTADORES

Sob um ponto de vista mais amplo, um sistema operacional pode ser considerado como sendo um conjunto de procedimentos manuais e automáticos que permitem a um grupo de pessoas compartilhar eficientemente os recursos de um computador (3). Se considerada essa definição, dentre a variedade dos aspectos envolvidos na análise e projeto de um sistema operacional, destacam-se os procedimentos automáticos destinados ao controle dos recursos de "HARDWARE" e "SOFTWARE" disponíveis no computador, atribuição desses recursos entre os vários programas usuários e ao estabelecimento de uma interface entre o "HARDWARE" e o "SOFTWARE". Estes procedimentos são nomalmente atribuidos ao "SOFTWARE EXECUTIVO" (7), isto é, ao conjunto de programas do Sistema Operacional encarregado desta tarefa.

Em se tratando de minicomputadores, o "SOFTWARE EXECUTIVO" assume uma importância considerável dentro do sistema operacional, podendo mesmo ser confundido com este. Neste caso, a natureza limitada dos recursos disponíveis e a variedade de aplicações dadas aos minicomputadores têm acarretado o surgimento de alguns tipos de "SOFTWARE EXECUTIVO", orientados de acordo com a natureza desses recursos e com a sua aplicaçao mais imediata.

De acordo com a referência (4), os sistemas operacionais para minicomputadores podem ser classificados em: Sistemas Operacionais "STAND-ALONE" ou Sistema Básico de Controle, Sistema Operacional com Disco (DOS-DISC OPERATING SYSTEM), Sistema Operacional de Tempo Real (RTOS - "REAL-TIME DISC OPERATING SYSTEM").

Algumas características mais importantes com respeito ao "SOFTWARE EXECUTIVO" associado a cada um destes sistemas serão assinaladas a seguir.

1.1.1 - SISTEMA BÁSICO DE CONTROLE (SBC)

Às vezes chamado de "Sistema Operacional de Fita de Papel" (13), um sistema "STAND-ALONE" dispõem de um conjunto de dispositivos de ENTRADA/SAÍDA do tipo teleimpressora (TTY), leitora e perfuradora de fita de papel e eventualmente impressoras de linhas e unidades de fita magnética. Estes dispositivos sao utilizados por programas dedicados a tarefas simples, como por exemplo, edição de textos, conversao de dados, etc..

O controle destes dispositivos requer normalmente a assistência de programas especiais (Volantes ou "DRIVERS" de Entrada/Saída) cuja complexidade depende da existência ou não de processadores de Entrada/Saída capazes de automatizar por "HARDWARE" a maioria das operações envolvidas no controle destes dispositivos.

Além dessa assistência, a facilidade de programação dessas operações, mesmo com a inclusão de novos dispositivos, constituem as principais funções do "SOFTWARE EXECUTIVO".

Um Sistema Básico de Controle pode servir de ponto de partida no desenvolvimento de outros sistemas operacionais se incorporar, além das funções de Entrada/Saída, outros procedimentos básicos necessários no estabelecimento da interface entre o "HARDWARE" e o "SOFTWARE".

1.1.2 - SISTEMA OPERACIONAL COM DISCO

A inclusão de discos magnéticos com a capacidade de armazenar grandes quantidades de dados, acessíveis de modo aleatório, proporciona um novo tipo de recurso ao sistema que pode ser controlado pelo "SOFTWARE EXECUTIVO". A facilidade com que os programas usuários podem manipular esses dados é representada pelo sistema de arquivos (FILE SYSTEM) propiciado nos Sistemas Operacionais com Disco (14).

A existência de um sistema de arquivos em disco propor ciona o recurso da "Segmentaçao de Programas", isto é, a possibi lidade de se executar programas que ocupam no total, uma área de armazenamento maior que a capacidade da memória principal. Com este recurso, através da seqüenciaçao dos vários segmentos que constituem um programa, o usuário pode utilizar comodamente os programas de desenvolvimento, em geral constituidos por vários passos, tais como, MONTADOR (Assembler), COMPILADORES, EDITORES, RELOCADOR-LIGADOR, etc, na confecção de seus próprios programas de aplicaçao. A sequenciação dos segmentos, isto é, a troca en tre segmentos residentes na memória principal por outros segmen tos do arquivo de programas no disco (Swapping Systems) (7) cons titui-se numa das principais atividades do "SOFTWARE EXECUTIVO" em um Sistema Operacional com Disco.

Neste tipo de sistema, a atribuição dos recursos é rea lizada sequencialmente entre os diversos usuários, de tal forma que a totalidade dos recursos disponíveis é atribuida ao único usuário em cada instante, dependendo apenas de sua requisição a través de diretivas enviadas ao "EXECUTIVO" ou "MONITOR".

1.1.3 - SISTEMAS OPERACIONAIS DE TEMPO REAL

Nas aplicações que envolvem outros tipos de dispositi vos de ENTRADA/SAÍDA, como por exemplo, conversores análogo--digitais e digitais-analógicos em um controle de processos ou aquisição de dados (6), a velocidade de resposta a estímulos ex ternos (interrupções) provocadas por esses dispositivos, torna--se um parâmetro importante no projeto do "SOFTWARE EXECUTIVO".

Normalmente esta "resposta" consiste na execução de um determinado programa, escolhido de acordo com o estímulo; entre os vários programas residentes na memória principal.

Dada a natureza assíncrona destes estímulos, o "EXECUTI VO" deve tomar as providências necessárias na ocorrência de con

flitos, por exemplo, estímulos simultâneos que exijam a execução de diferentes programas naquele instante. Se a cada programa estiver associada uma prioridade, o "EXECUTIVO" deve resolver esses conflitos baseado neste esquema de prioridade.

Um outro tipo de dispositivo, chamado RELÓGIO DE TEMPO REAL, permite ainda que programas possam ser executados periodicamente ou em instantes determinados. De um modo geral, a escolha do próximo programa a ser executado (Scheduling) e a comunicação entre um operador e os programas em questão, constituem as principais tarefas de um "SOFTWARE EXECUTIVO" em um sistema Operacional de Tempo Real.

1.1.4 - SISTEMAS OPERACIONAIS DE TEMPO REAL COM DISCO

As características de um Sistema Operacional de Tempo Real e de um Sistema Operacional com Disco, mencionadas anteriormente, podem estar associadas em um único sistema de maior capacidade, chamado Sistema Operacional de Tempo Real com Disco.

A não ser em uma aplicação, por exemplo, de aquisição de dados na qual o disco é utilizado para coletar esses dados por um período relativamente longo, as tarefas em "Tempo Real" são independentes das outras relacionadas com a confecção de um programa.

Essa características de "MULTIPROGRAMAÇÃO" introduz outros requisitos ao "SOFTWARE EXECUTIVO", quais sejam, as de garantir a não interferência entre os programas executados em "Tempo Real" (FOREGROUND) e aqueles de menor prioridade (BACKGROUND) concorrendo com os recursos disponíveis no sistema.

1.2 - IMPLEMENTAÇÃO DE UM SISTEMA BÁSICO DE CONTROLE (S.B.C.)

Os principais aspectos da implementação de um sistema operacional simples do tipo "STAND-ALONE" desenvolvido no minicomputador G-10 (12) são abordados neste ítem, com a finalidade

de ilustrar algumas técnicas que serão utilizadas na implementação de um sistema do mesmo tipo para o minicomputador "PATINHO FEIO".

Este sistema denominado SISTEMA BÁSICO DE CONTROLE foi projetado com as seguintes finalidades:

a) uniformizar e facilitar a programação das operações de ENTRADA/SAÍDA;

b) tratar as interrupções especiais;

c) facilitar o teste de um conjunto de programas ligados ao "SOFTWARE BÁSICO" desenvolvido para o G-10, tais como, O MONTADOR (16), CARREGADOR-LIGADOR (15) e um Compilador FORTRAN, utilizando o próprio G-10.

Este sistema foi desenvolvido adotando-se uma configuração que inclui uma memória com 16K palavras de 16 bits, leitora e perfuradora de fita de papel, uma teleimpressora (TTY), leitora de cartões e uma impressora de linha.

1.2.1 - CARACTERÍSTICAS ESSENCIAIS DO MINICOMPUTADOR G-10

As principais características do minicomputador G-10 descrito na referência (12), e que são essenciais no desenvolvimento de um Sistema Básico de Controle com os propóstios citados, podem ser resumidas da seguinte forma:

a) ENDEREÇAMENTO DA MEMÓRIA PRINCIPAL

A principal característica em relação ao endereçamento é a existência de endereçamento relativo com respeito a duas bases, a base de programas e a base de dados. Deste fato decorre a divisão de um programa em duas áreas lógicas, a área de códigos e a área de dados, referenciados por meio de instruções distintas.

Cada uma dessas áreas é limitada por um registrador LIMITE DE PROGRAMA E LIMITE DE DADOS respectivamente.

b) MODOS DE OPERAÇÃO

Dois modos de operaçao da Unidade Central de Processamento, chamados MODO SUPERVISOR e MODO USUÁRIO permitem que um conjunto de instruções especiais, constituidas por instruções do tipo, mudanças de bases, instruçoes de ENTRADA/SAÍDA, etc.., seja executada somente no MODO SUPERVISOR. Em particular, neste modo, a base de programas é feita automaticamente igual a zero enquanto que a base de dados e limites podem admitir quaisquer valores.

c) OPERAÇOES DE ENTRADA/SAÍDA

Os dispositivos de ENTRADA/SAÍDA conectados no CANAL CONCENTRADOR, são controlados diretamente pela U.C.P., através de instruçoes de ENTRADA/SAÍDA executadas na transferência de cada dado destes dispositivos. Exceto a impressora de linha, os demais dispositivos utilizadores no S.B.C. estão conectados no CANAL CONCENTRADOR.

A impressora de linha está conectada no CANAL SELETOR , o qual tem a capacidade de realizar a transferência de um bloco de dados com acesso direto à memória principal, interagindo com a U.C.P. somente no início e final da transferência do bloco de dados.

d) INTERRUPÇOES

Qualquer interrupçao da U.C.P. é caracterizada por uma mudança para MODO SUPERVISOR, armazenamento do CONTADOR DE INSTRUÇOES (Registrador R0), BASE DE PROGRAMAS E PALAVRA DE "STATUS" nas três posiçoes de memória consecutivas a partir daquela apontada pelo registrador R1(REGISTRADOR TOPO DA PILHA). Após estas operaçoes, o CONTADOR DE INSTRUÇOES é carregado com o valor contido em uma posição fixa da memória de acordo com a origem da interrupçao.

As interrupçoes podem ser provocadas pelos canais de ENTRADA/SAÍDA, pela execuçao da instrução CSUP (chamada de "SUPERVISOR") ou por eventos especiais tais como violação de memória, execuçao de instruçao privilegiada em MODO USUÁRIO, falha de alimentação ou por meio do painel.

1.2.2 - ORGANIZAÇAO DO SISTEMA BÁSICO DE CONTROLE DO G-10

O sistema Básico de Controle é constituido por um conjunto de rotinas executadas com MODO SUPERVISOR e supõe a existência de apenas um programa executado em MODO USUÁRIO.

A distribuição dos códigos e dados do S.B.C. e do programa usuário na memória principal sao feitos de acordo com o esquema da fig. 1.1.

A área POSIÇOES DE INTERRUPÇAO (0 a 80) é reservada para conter os endereços das rotinas de tratamento das interrupções, conforme o esquema de interrupçao adotado.

O S.B.C. opera com base de programa (B P S) igual a zero por ser executado em MODO SUPERVISOR. A utilizaçáo da base de dados (B D S) também igual a zero permite que o S.B.C. tenha acesso a toda memória como se esta fosse uma única área de dados.

Os dados propriamente ditos do S.B.C. foram reunidos nas primeiras posiçoes que seguem as POSIÇOES DE INTERRUPÇAO (ÁREA DE DADOS DO S.B.C.) enquanto que os códigos estao agrupados na ÁREA DE CÓDIGOS DO S.B.C.

Os dados referenciados através do registrador R1 (PILHA) manipulados pelo S.B.C., por seu caráter dinâmico, isto é, por não ser previsível exatamente a quantidade de posições que poderao ocupar, foi colocada nas últimas posiçoes de memória(3 F 50), sendo indicadas pela variável R1S (R1 do Supervisor).

A possibilidade do S.B.C ter acesso a totalidade da memória faz com que o registrador LIMITE DE DADOS assuma a indica

DISTRIBUIÇÃO DA MEMÓRIA NO S.B.C. DO G-10

Limite	Área
BPS,BDS	POSIÇÕES DE INTERRUPÇÃO
	AREA DE DADOS DO S.B.C
	AREA DE CODIGOS DO S.B.C
BPU	AREA DE CODIGOS DO USUARIO
LPU	DESCRITOR
BDU	AREA DE DADOS DO USUARIO
R1I	PILHA DO USUARIO
LDU	AREA LIVRE
R1S	PILHA DO S.B.C
LDS	

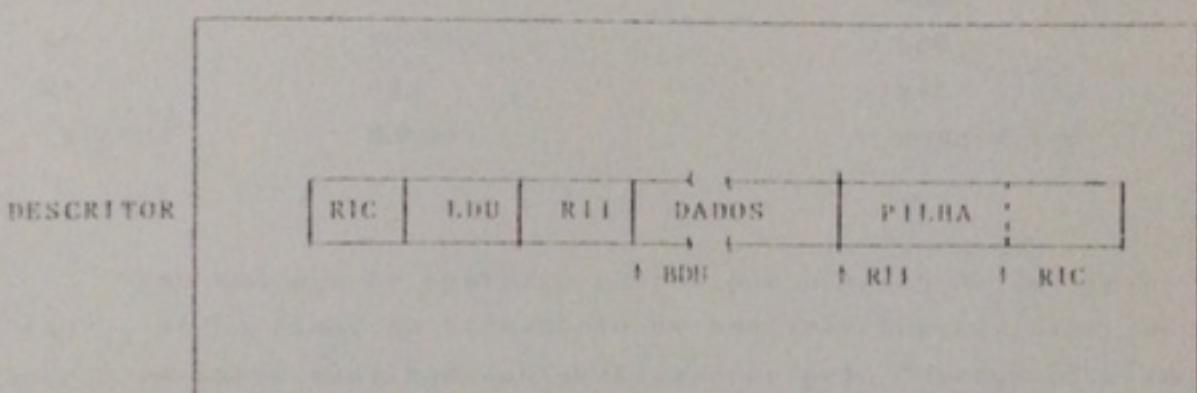



Fig. 1.1

ção da última posição de memória disponível (L D S) quando as rotinas do S.B.C. estiverem em execução.

O programa usuário é constituído por ÁREA DE CÓDIGOS DO USUÁRIO delimitada por uma base de programa (BPU) e um limite de programa (LPU). A ÁREA DE DADOS DO USUÁRIO, por sua vez, é constituida por um DESCRITOR, DADOS E PILHA conforme ilustrado abaixo:

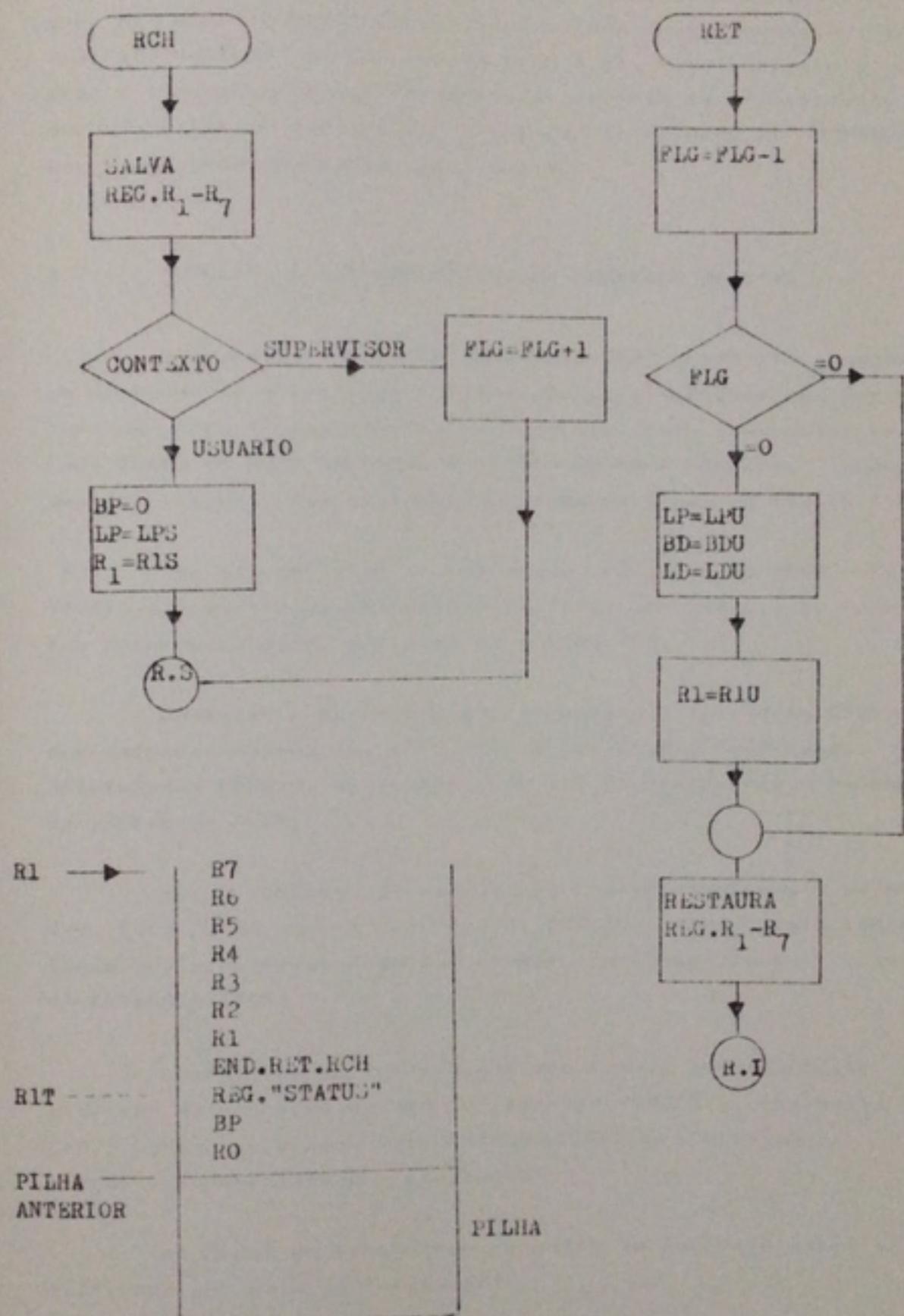
O DESCRITOR é constituído por três palavras, contendo os valores do APONTADOR DA PILHA (R I) CORRENTE (RIC), O LIMITE DE DADOS e o valor inicial do APONTADOR DA PILHA (RII).

Este DESCRITOR será utilizado na mudança de contexto.

1.2.3 - TRATAMENTO DE INTERRUPÇOES E MUDANÇA DE CONTEXTO

Um CONTEXTO pode ser caracterizado pelos valores dos registradores da U.C.P., BASE DE PROGRAMAS, LIMITE DE PROGRAMAS, BASE DE DADOS, LIMITE DE DADOS, APONTADOR DA PILHA (RI) e PALAVRA DE "STATUS". Na PALAVRA DE "STATUS" em particular, estao codificados os bits de MODO (Supervisor ou Usuário) e de interrupçao (I) da U.C.P. (liberada ou inibida).

Dois contextos são utilizados no S.B.C., o CONTEXTO SUPERVISOR e o CONTEXTO USUÁRIO, nos quais os valores desses registradores são tabelados abaixo:

	CONTEXTO SUPERVISOR	CONTEXTO USUÁRIO
BP	0	BPU
LP	LPS	LPU
BD	0	BDU
LD	$(FFFF)_{16}$	LDU
RI	RIS	RIC
"STATUS"	MODO=1	MODO=0,I=0

Uma mudança de contexto ocorre por ocasião de uma interrupção ou no final do tratamento de uma interrupção. Essa mudança é em parte realizada automaticamente pelo "Hardware" e em parte por duas rotinas especiais do S.B.C. chamadas RCH (rotina de mudança de contexto) e RET (Retorno de interrupção).

A rotina RCH, utilizando uma variável de controle FLG, realiza a mudança para CONTEXTO SUPERVISOR se o contexto anterior a uma interrupção for USUÁRIO, de acordo com o diagrama da figura 1.2.

O tratamento de qualquer interrupção inicia-se por uma chamada de rotina RCH. Nesta ocasião, a PILHA (Supervisor ou Usuário) terá a configuração indicada na figura 1.2., sendo R0, BP e "STATUS" armazenados pela interrupção, o endereço de retorno de RCH (END.RET RCH) pela instrução de chamada de subrotina (PUG RI, *I RCH) e os valores de R1 a R7 armazenados pela instrução de SALVA REGISTRADORES (SVR RI).

Pelo exame desse "STATUS" na PILHA, obtem-se o MODO anterior, o que vai determinar a mudança de contexto ou não. No caso de uma interrupção ocorrer no contexto SUPERVISOR, esta é apenas "contada" na variável FLG. Havendo mudança de contexto, o DESCRITOR do USUÁRIO é atualizado convenientemente.

DIAGRAMAS DAS ROTINAS RCH e RET

Fig. - 1.2

No final do tratamento da interrupção, a rotina RET examina por meio de FLC a necessidade ou não de uma mudança para contexto USUÁRIO. Se for o caso (FLC = 0), este contexto é assumido e finalmente o registradores do usuário são restaurados por meio da PILHA. É executada a instrução de RETORNO DE INTERRUPÇÃO com o apontador da PILHA igual a RIT.

1.2.4 - FUNÇÕES DO SISTEMA BÁSICO DE CONTROLE DO G-10

A instrução "CHAMADA DE SUPERVISOR" (CSUP n) comporta um operando de 8 bits que é utilizado para designar uma particular função do Sistema Básico de Controle. Esta instrução, executada tanto em MODO SUPERVISOR quanto em MODO USUÁRIO, provoca uma interrupção cujo tratamento é esquematizado na figura 1.3.

Uma vez que este tratamento é realizado em MODO SUPERVISOR, a primeira providência é realizar uma mudança de contexto, caso necessário, por meio da rotina RCH.

Conforme o parâmetro que acompanha a instrução CSUP, um determinado subprograma do S.B.C. é executado, realizando uma determinada FUNÇÃO, numeradas de 0 a 9 de acordo com o parâmetro da instrução CSUP.

Estas FUNÇÕES são executadas com interrupções permitidas, isto é, durante a execução da FUNÇÃO, outras interrupções podem ser atendidas, como por exemplo interrupções dos canais de ENTRADA/SAÍDA.

O processo de interrupção mencionado anteriormente permite que na execução de uma determinada FUNÇÃO i, uma outra FUNÇÃO j seja executada. Este fato possibilita a utilização de FUNÇÕES já anteriormente programadas.

No final do tratamento, a volta ao contexto anterior é realizada por meio da rotina RET.

TRATAMENTO DA INTERRUPÇÃO "CHAMADA DE SUPERVISOR" (CSUPn)

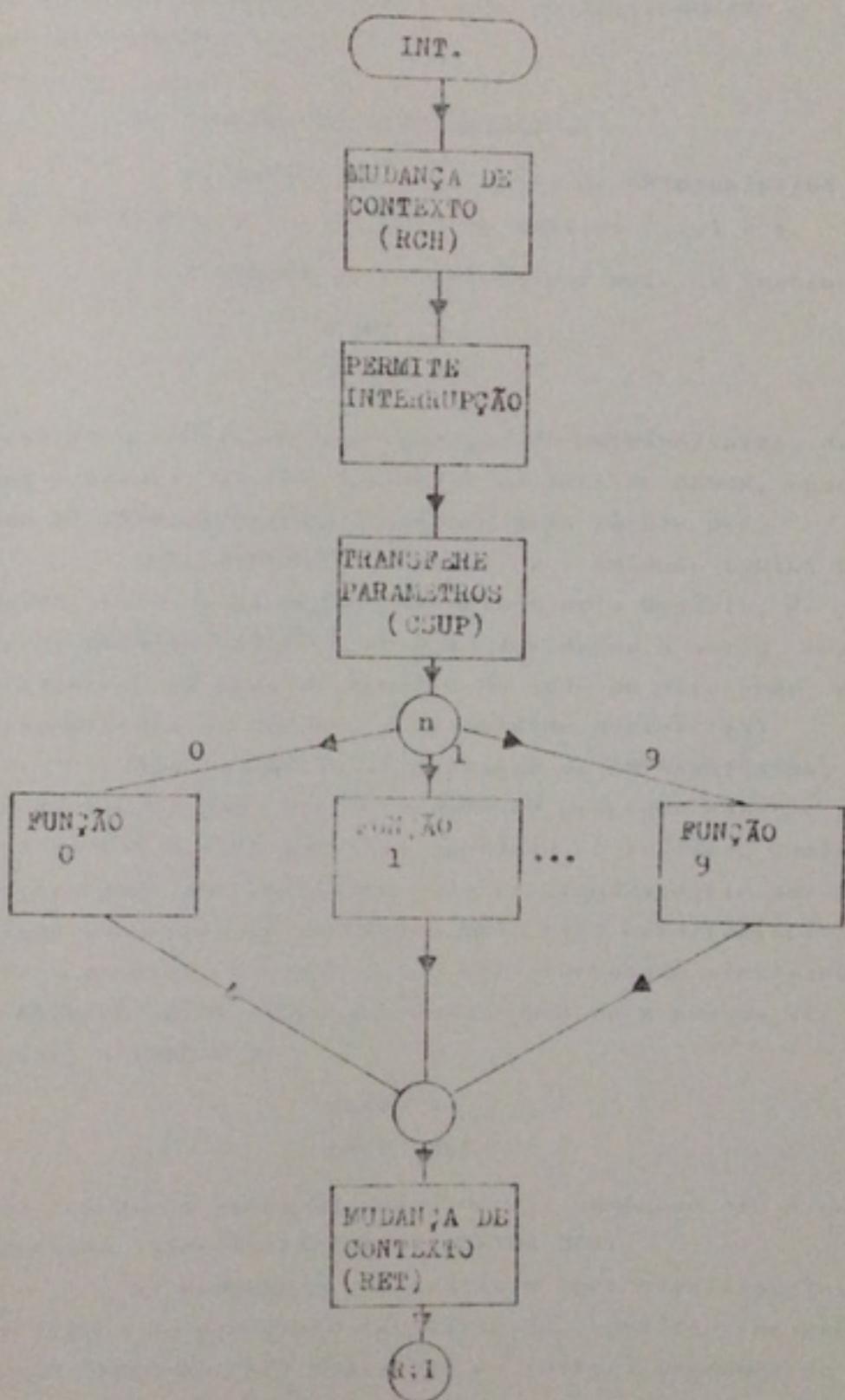



Fig. - 1.3

As dez FUNÇOES implementadas no Sistema Básico de Controle estao relacionadas com operaçoes de ENTRADA/SAÍDA controle do programa Usuário, controle do RASTREAMENTO e interpretaçao de comandos.

a) FUNÇOES DE ENTRADA/SAÍDA

As principais operaçoes de ENTRADA/SAÍDA de dados são realizadas pelas FUNÇOES de números 1,2,3 e 6.

A FUNÇÃO 1, executada por meio da instrução:

```
CSUP  1
DFCP  DES
```

realiza o início de uma operaçao de ENTRADA/SAÍDA, descrita por uma sequência de três palavras da ÁREA DE DADOS, chamada DESCRITOR DE ENTRADA/SAÍDA, designada pelo rótulo DES.

Este DESCRITOR especifica a unidade LOGICA DE ENTRADA/SAÍDA, isto é, um número associado pelo Usuário, do dispositivo de ENTRADA/SAÍDA, o formato dos dados a serem transmitidos, o endereço da área de memória de onde ou para onde deverão ser transmitidos os dados e a quantidade destes (15).

Após o início da operação de ENTRADA/SAÍDA, o controle da U.C.P. pode ser retornado ao programa Usuário ou permanecer no S.B.C. até que esta operaçao se realize. Qualquer das opçoes pode ser escolhida pelo programa usuário por meio de códigos especiais no DESCRITOR DES. Esta característica permite que o programa usuário possa ser processado simultaneamente com a ENTRADA/SAÍDA. Para que essas operaçoes possam ser sincronizadas, a FUNÇAO 3.

```
CSUP  3
DFCP  DES
```

faz com que o programa usuário seja suspenso até a conclusao da operaçao especificada no DESCRITOR DES.

As FUNÇOES 2 e 6 realizam operaçoes especiais nos dispositivos de perfuraçao de fitas de papel (perfuraçao de alimentaçao "FEED-HOLES") e leitora de cartoes (mudança de escaminho).

b) FUNÇOES DE CONTROLE DO PROGRAMA USUÁRIO

Para o controle do programa usuário, as FUNÇOES 0, 5 e 4, realizam o término do programa usuário (CSUP 0 ou CSUP 5) ou a suspensao temporária do programa até que um comando especial (V) seja enviado pela teleimpressora (CSUP 4).

c) FUNÇOES DE CONTROLE DO RASTREAMENTO

Uma característica do minicomputador G-10 é a existência de um tipo de interrupçao, chamado RASTREAMENTO ("TRACE"), que pode ocorrer antes do início da execuçao de cada instruçao pela U.C.P.

O tratamento desta interrupçao, no S.B.C., é realizado por meio da impressão em um dispositivo designado pelo usuário, do conteúdo dos registradores do U.C.P. Deste modo um programa pode ser acompanhado durante sua execuçao, facilitando o seu teste.

O controle dos trechos do programa nos quais este rastreamento deve ser realizado, é obtido por meio das funções de "LIGA RASTREAMENTO" (CSUP 7) e "DESLIGA RASTREAMENTO" (CSUP 8).

d) INTERPRETAÇÃO de COMANDOS

A FUNÇAO 9 permite que o S.B.C. execute outras funçoes resultantes da interpretaçao de um comando designado por essa FUNÇAO.

Assim, a intruçao:

DFCP COM

causa a interpretaçao do comando codificado na área de dados indicada pelo rótulo COM.

Um conjunto de tais comandos, implemen[illegible] neste S.B.C. serao descritos em 1.2.[illegible].

1.2.5 - COMANDOS DO SISTEMA BÁSICO DE CONTROLE

A interação entre o Sistema Básico de Controle e o operador é estabelecida por meio de COMANDOS e MENSAGENS transmitidas por meio de uma teleimpressora (TTY).

Esta interação caracteriza o S.B.C., cujo diagrama resumo está esquematizado na fig. 1.4.

Esse conjunto de comandos, resumidos na tabela abaixo, representam as principais operações necessárias na condução do teste e execução de um programa.

As MENSAGENS, para "COMANDO NÃO RECONHECIDO" através do sinal (?) e "COMANDO REALIZADO" (@) permitem que esta interação se complete.

TABELA DE COMANDOS (26)

COMANDO	SIGNIFICADO
A	Termina programa em execução.
B	Lista Bases e Limites do programa.
C,N,X	Carrega programa da unidade N e reserva X posições para a pilha.
EP,E,lista	Modifica as posições da área de programas a partir do endereço E com os valores da lista.
IP, lista	Acrescenta os valores constantes da lista na área de programas.
ID, lista	Acrescenta os valores constantes da lista na área de dados.
LP,N,X,Y	Lista na unidade N a área de programas entre os endereços X e Y.

LD,N,X,Y	Lista na unidade N a área de dados entre os endereços X e Y.
P, E	Suspende a execuçao do programa quando este executar a instruçao de endereço E.
R	Executa o programa usuário.
T,N,X,Y	Realiza rastreamento na unidade lógica N na regiao de endereços X a Y.
U,X,Y	Troca unidades lógica X por Y.
V	Continua execuçao do programa.

1.3 - ASPECTOS DO PROJETO DE UM SISTEMA OPERACIONAL

A extensao e os tipos de funçoes a serem implementadas em um sistema operacional sofrem uma grande variaçao, desde um sistema operacional simples como um SISTEMA BÁSICO DE CONTROLE até um Sistema Multiprogramado como o de um SISTEMA DE TEMPO-REAL COM DISCO.

Entretanto, o principal problema no projeto de um sistema operacional nao é justamente definir as funçoes que satisfaçam a dadas especificaçoes; mas antes, definir um Núcleo básico de programaçao que possa ser expandido de acordo com as necessidades do sistema (3).

Principalmente quando se trata de minicomputadores, essa capacidade de expançao torna-se muito importante uma vez que pode ocorrer em paralelo com a inclusao de novos recursos de "Hardware" , como por exemplo, de um disco magnético. Neste caso em particular, como pode ser notado na classificaçao dos sistemas operacionais para minicomputadores, a inclusao do disco possibilita uma grande expansao na capacidade desses sistemas.

Juntamente com a expansao, a facilidade com que um dado sistema operacional possa ser modificado, possibilitando sua otimizaçao de acordo com as particularidades de uma aplicação .

DIAGRAMA RESUMO DO SISTEMA BÁSICO DE CONTROLE (G-10)

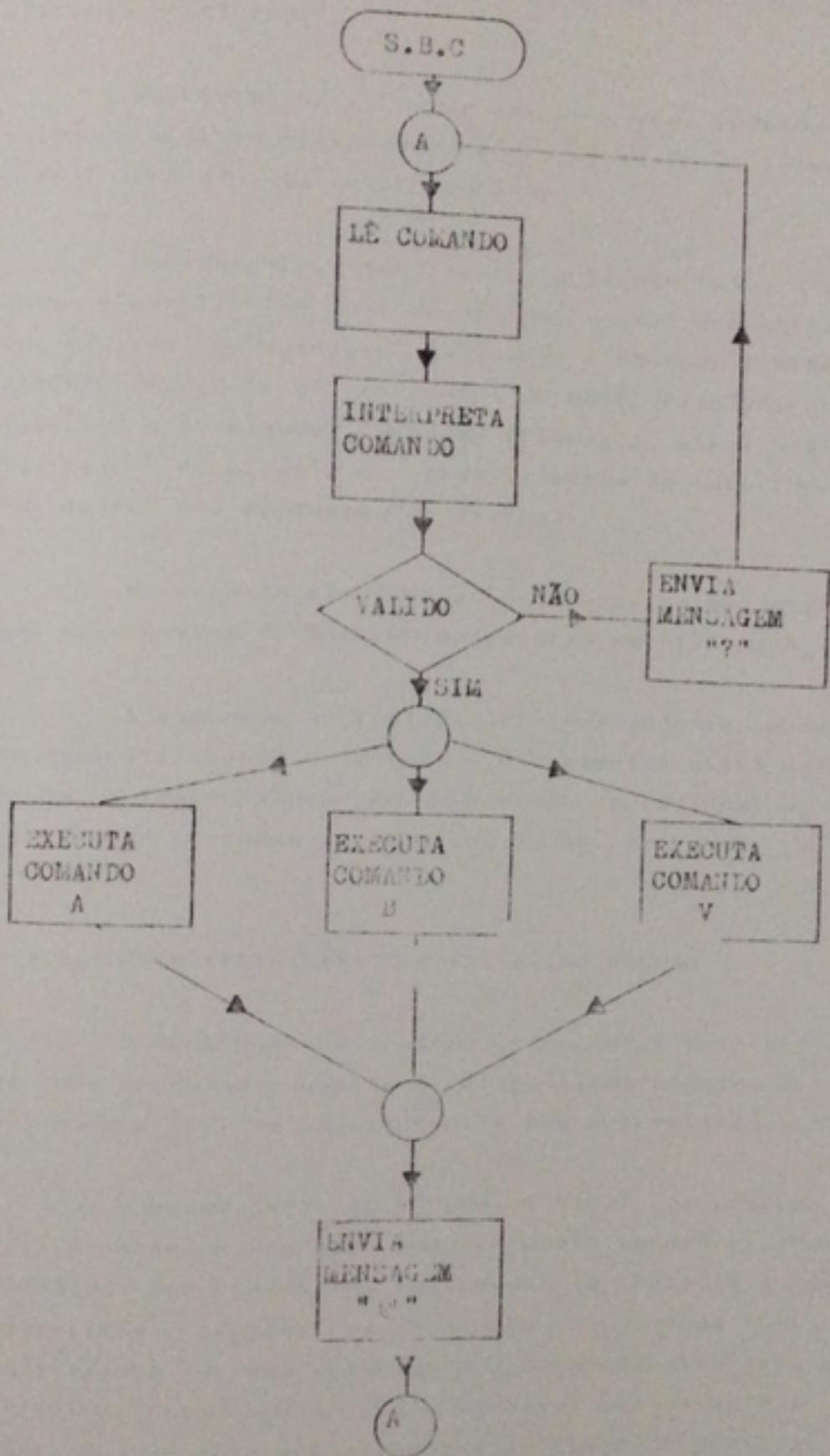
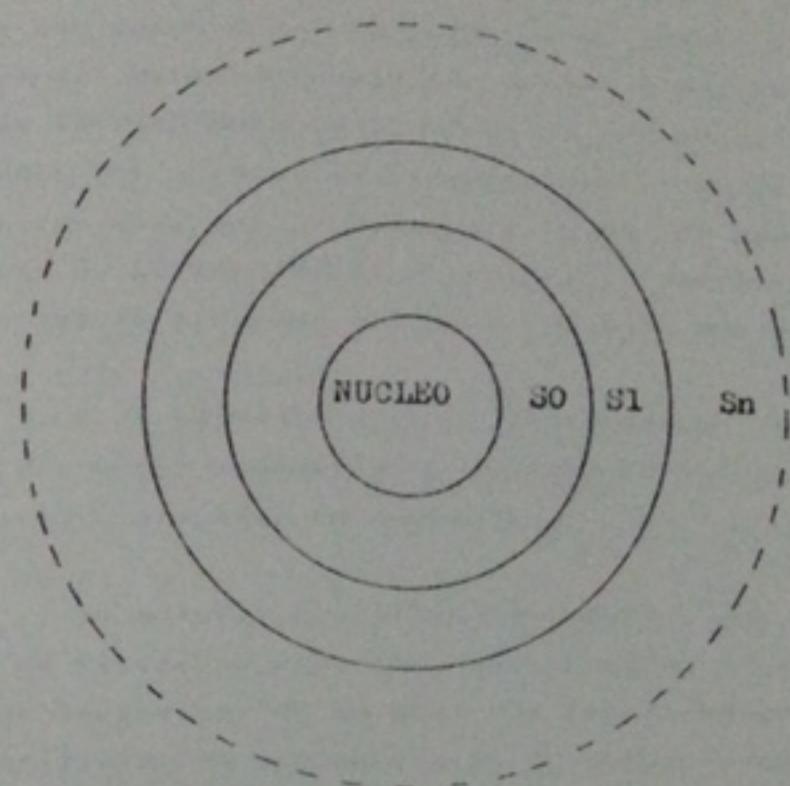



Fig. - 1.4

conduzem à necessidade de se ter uma implementação com caracte rísticas modulares.

Uma forma de se organizar os vários módulos que constituirão o sistema operacional pode ser vista na referência (10) a qual será adotada neste trabalho.

Basicamente, como ilustra a figura 1.5., o primeiro módulo é constituido por um NÚCLEO, o qual concentra os recursos básicos de "Hardware" tais como a UNIDADE CENTRAL de PROCESSAMENTO (U.C.P.), processadores de ENTRADA/SAÍDA, conjunto de instruções de máquina, memória principal, etc., além do "Software" necessário no estabelecimento da interface entre o "Hardware" e o restante do sistema.

No segundo nível estão os processos que utilizando apenas os recursos do Núcleo implementam um sistema S_0.

A expansão se realiza acrescentando-se novos níveis, cada qual utilizando os recursos do primeiro nível mais interno. Assim, a implementação de um sistema operacional S_n é realizada utilizando-se todos os recursos de um sistema S_{n-1}.

1.3.1 - PRINCIPAIS CARACTERÍSTICAS DO NÚCLEO

O NÚCLEO de um sistema operacional pode ser caracterizado como sendo um conjunto de dispostivos capazes de executar uma série de ações, em uma sequência bem determinada.

Fazendo parte do NÚCLEO, a U.C.P. executa as instruções de máquina enquanto possivelmente um processador de ENTRADA/SAÍDA executa outras instruções no controle de um ou mais dispositivos periféricos. Nos dois casos, cada instrução é caracterizada por uma série de ações em uma sequência previamente definida. Estas ações podem envolver por exemplo a soma dos conteudos de dois registradores do FLUXO DE DADOS ou o envio de um sinal de comando para um dispositivo de ENTRADA/SAÍDA.

ORGANIZAÇÃO DE UM SISTEMA OPERACIONAL

Fig. - 1.5

Na execuçao de um programa, ou seja, na execuçao da sé rie de açoes determinadas pela sequência de instruçoes de má quina que compoem tal programa, esta sequência pode nao estar cronologicamente determinada, devido a possibilidade de ocorrência de INTERRUPÇOES (8). Essas interrupçoes sao normalmente causadas por eventos assíncronos com a execuçao das instruçoes e sao devidos, por exemplo, a términos de operaçoes em dispo sitivos de ENTRADA/SAÍDA ou condiçoes anormais resultantes da tentativa do programa utilizar recursos nao autorizados.

O atendimento dessas interrupçoes é normalmente condu zido por outra sequência de instruçoes nao diretamente relacio nada com o programa em andamento.

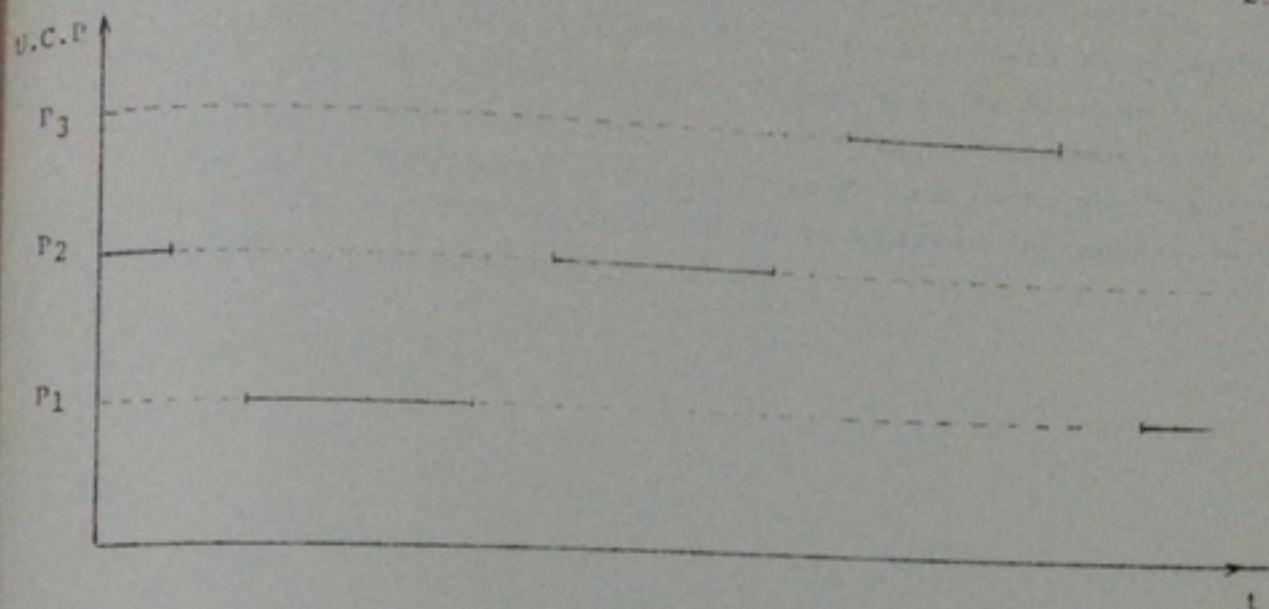
Na maioria dos minicomputadores, sao previstas instru çoes ou situaçoes especiais que permitem a execuçao de uma sé rie de instruçoes de um modo nao interrompivel. Este fato pode ser utilizado na implementaçao de subprogramas nao interrompí veis chamados PRIMITIVOS.

Um PRIMITIVO se processa como se fosse uma instruçao de máquina, possibilitando por esse motivo, uma ampliaçao da capa cidade de processamento do NÚCLEO através do emprego de um con junto adequado de PRIMITIVOS.

Do ponto de vista de um sistema do tipo SO, o NÚCLEO se apresenta portanto como uma "MÁQUINA VIRTUAL" (10) com capaci dade superior à do "Hardware" que existe efetivamente.

1.3.2 - O CONCEITO DE "PROCESSO"

Um programa pode ser caracterizado como sendo a descri çao, por meio de códigos da linguagem de máquina, de uma série de açoes a serem realizadas pela U.C.P. Uma vez localizado em alguma área da memória principal, para que este programa possa ser executado, é necessário que a U.C.P. assuma um determinado "SATATUS" que pode ser definido pelos valores dos registradores de

propóstio geral, CONTADOR DE INSTRUÇÕES, conteudo de algum registrador de "STATUS" do FLUXO de DADOS, Modo de operaçao da U.C.P., etc., dependendo da particular U.C.P. À medida que esta executa as instruçoes indicadas pelos códigos do programa, vai assumindo novos "STATUS" que são funçao do "STATUS" anterior e de cada instruçao executada.

Uma vez que tal sequência de instruçoes pode ser interrompida, alguma providência deve ser tomada no sentido de se salvar esse "STATUS" quando ocorrer a interrupçao, para que o programa em execuçao possa ser retomado corretamente após o atendimento da interrupçao. Associando-se o valor do "STATUS" da U.C.P. em cada instante ao programa em execuçao, pode-se conceituar um PROCESSO como sendo uma sequência de açoes A_0, A_1,..... A_i, A_{i+1},..... e uma sequência de estados S_0, S_1,...S_i, S_{i+1}... de um programa em execuçao, de tal forma que um estado S_{i+1} é determinado por A_{i+1}, e a açao A_{i+1} é funçao do estado S_i (5).

Uma vez que um processo irá constituir-se no elemento fundamental na implementaçao de um sistema operacional, é necessário que em cada caso seja dada uma definiçao precisa e objetiva para este conceito.

1.3.3 - MULTIPROCESSOS

Uma vez que um processo pode ser interrompido e reassumido posteriormente, a técnica de MULTIPROCESSOS (MULTITASKING) é frequentemente utilizada na implementaçao de um Sistema Operacional. Para isto, o sistema admite a existência de vários processos simultâneos utilizando os recursos do sistema, daí o termo "MULTIPROCESSOS".

Desse modo, a U.C.P. é cedida a cada um dos processos e estes a utilizam por um determinado período de tempo, até que algum evento ocasione a interrupçao deste processo. O diagrama abaixo ilustra as atividades da U.C.P. de acordo com sua utilizaçao por três processos P_1, P_2 e P_3.

Pode-se observar que a comutação da U.C.P. entre um processo e outro não ocorre instantaneamente, exigindo um certo intervalo de tempo ("OVERHEAD") no salvamento do "STATUS" associado a P_1 e a recuperação do "STATUS" de P_2, por exemplo, no qual a U.C.P. é utilizada para isto.

A técnica de MULTIPROCESSOS visa otimizar o tempo de uso da U.C.P., uma vez que normalmente um processo pode ficar bloqueado aguardando, por exemplo, a execução de uma operação de ENTRADA/SAIDA (E/S). Enquanto isso, pode existir algum outro processo independente em condições de utilizar a U.C.P.

Em outros casos, a utilização de MULTIPROCESSOS decorre naturalmente do tipo de aplicação do Sistema Operacional, como é o caso de um Sistema de Tempo-Real.

Para que um Sistema Operacional realiza as funções para as quais está destinado, utilizando a técnica de MULTPROCESSOS, será necessário que leve em conta a capacidade de sincronização e comunicação entre os vários processos e a possibilidade de adição e remoção de processos de um modo dinâmico (3).

Quando se tem características de MULTIPROGRAMAÇAO, como nos Sistemas de Tempo-Real com Disco, a existência de processos concorrentes (5) com os recursos disponíveis no Sistema ocasiona a necessidade da inclusao de processos relacionados com a atribuição e controle desses recursos de uma forma que possa garantir a não interferência entre as atividades independentes do mesmo Sistema Operacional.

CAPÍTULO 2 - DEFINIÇÃO E IMPLEMENTAÇÃO DE UM NÚCLEO NO "PATINHO FEIO"

2. DEFINIÇÃO E IMPLEMENTAÇÃO DE UM NÚCLEO NO "PATINHO FEIO"

Conceitualmente, o NÚCLEO será constituido por uma coleção de programas e dispositivos do "PATINHO FEIO" capazes de implementar um determinado conjunto de ações denominados PRIMITIVOS. Esses primitivos incluirão o conjunto de instruções de máquina do "Patinho Feio" e outros que serão relacionados neste capítulo. Inicialmente, com a finalidade de caracterizar objetivamente o conceito de primitivo, alguns detalhes do "Hardware" do "Patinho Feio" serão evidenciados.

2.1 - O ESQUEMA DE INTERRUPÇÃO DO "PATINHO FEIO"

Os detalhes do esquema de interrupção no "Patinho Feio" estão definidos nas referências (1) e (2). Um modelo simplificado deste esquema será adotado aqui para efeito de referência, lembrando apenas seus aspectos funcionais, (fig. 2.1).

Existe a possibilidade de se conectar até 15 interface à U.C.P., em posições numeradas de 1 a 15. Em cada interface j, existe um conjunto de sinais que participam da lógica de interrupção dos quais se destacam os sinais ESTADO (ESTj), PERMITE INTERRUPÇÃO DA INTERFACE (PRI j) e PEDIDO DE INTERRUPÇÃO DA INTERFACE (PDIj).

Estes sinais estão relacionados por: $(PDI)j=(EST)j.(PRI)j$. Assim a interface enviará um pedido de interrupção à U.C.P. sempre que o ESTADO assumir nível 1 e sua interrupção estiver permitida ($(PRI)j=1$). Estes sinais podem ser alterados por instruções executadas na U.C.P.

Os sinais de pedido de interrupção são enviados à U.C.P. e concentrados no bloco lógico A que através de uma lógica "OU" produz o sinal de INTERRUPÇÃO PENDENTE (INP). O sinal INP não depende de nenhuma outra condição, ou seja,
$(INP)=(PDI)_1 + \ldots + (PDI)_j + \ldots (PDI)_{15}$. Esse esquema não pressupõe nenhuma prioridade com relação aos pedidos de interrupção de forma que nesse estágio existirá apenas uma interrupção pendente

SISTEMA DE INTERRUPÇÃO DO "PATINHO FEIO"

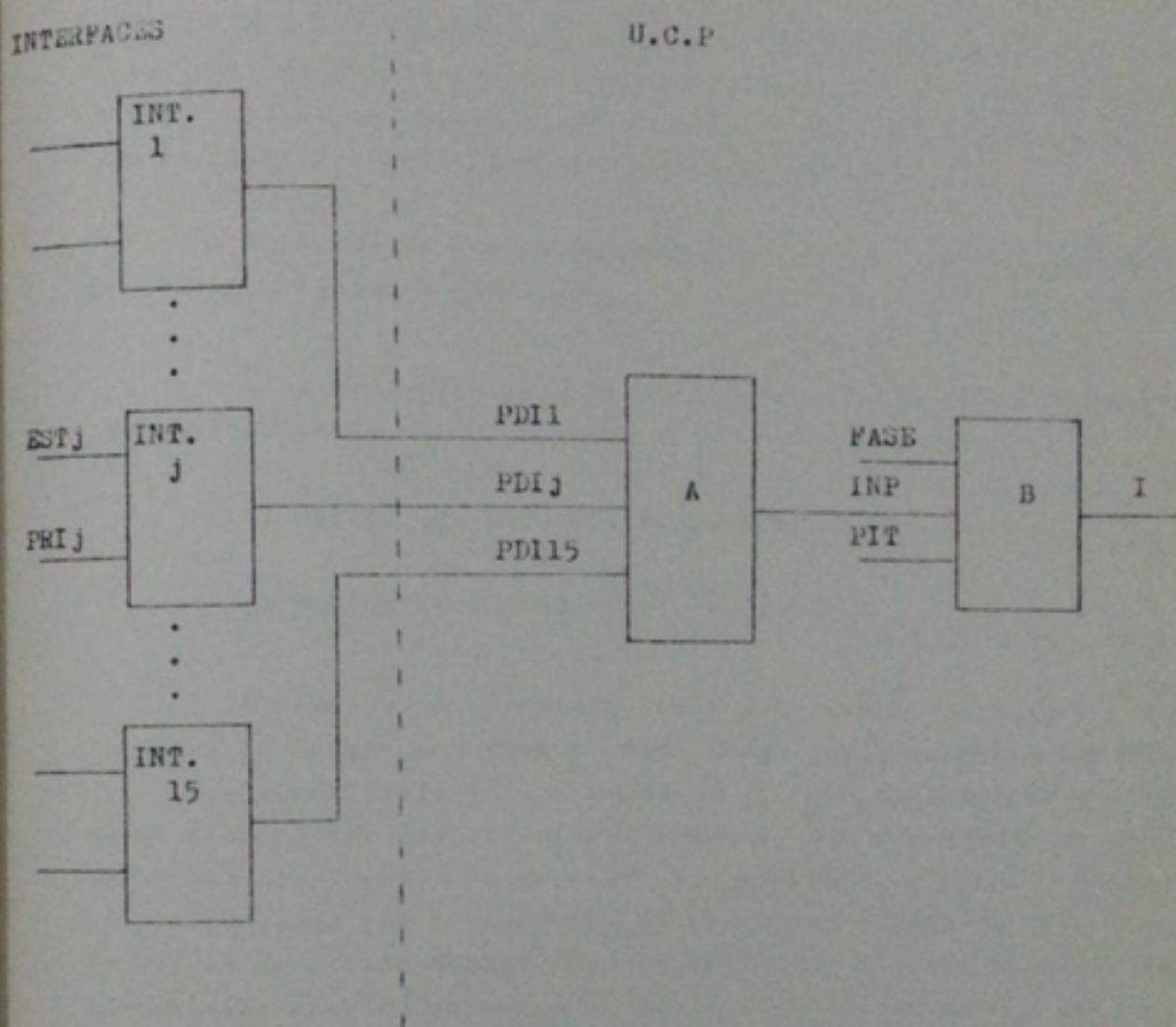



Fig. - 2.1

quaisquer que sejam as interfaces que enviarem pedidos de inter rupçao pendente simultaneos.

A U.C.P. executa as instruçoes provenientes da memória principal, em dois estados: NORMAL((I) = 0) e INTERROMPIDO ((I) = 1). Esses estados sao caracterizados pelos valores lógi cos do sinal I ("FLIP-FLOP" de interrupçao), ref. (I).

A transição do estado NORMAL para o estado INTERROMPIDO ocorre nas seguintes condições:

a) A transiçao está permitida através do sinal PERMITE INTERRUPÇAO (PIT), ou seja (PIT) = 1.

b) Existe uma interrupção pendente (INP) = 1.

c) O sinal FASE tem nível lógico 1.

A condição a) é estabelecida pela U.C.P. através da execuçao de instruçoes PERM e INIB, sendo que a primeira faz (PIT) = 1 e a segunda (PIT) = 0. A condiçao b) está relacionada com eventos ocorridos nas interfaces enquanto que a condiçao c) está associada ao processo de execuçao das instruções. Esta última foi introduzida com o propósito de ressaltar o fato de que a transição do estado NORMAL para INTERROMPIDO só pode ocorrer em determinada fase da execuçao de uma instruçao, ou mais precisa mente, no início da fase de busca da próxima instruçao (ref.(I)) Assim, durante a execução de uma instruçao nao existe a possibi- lidade de ocorrer uma transiçao deste tipo.

Essa transiçao, quando ocorrer, é acompanhada por duas providências tomadas pela U.C.P. Em primeiro lugar, o CONTADOR DE INSTRUÇOES (CI), isto é, registrador da U.C.P. que contém o ende reço da memória da próxima instruçao a ser executada, é armaze nado nas posiçoes de endereço 2 e 3 da memória. Em segundo lugar, o CONTADOR DE INSTRUÇOES recebe o valor 4 iniciando assim a exe cuçao da instruçao contida nesta posiçao de memória.

A transição do estado INTERROMPIDO para o estado NORMAL somente é realizada por meio da execução de uma instrução especial denominada PUL. Esta instrução faz com que o CONTADOR DE INSTRUÇÕES receba o valor contido nas posições de endereços 2 e 3 da memória, proporcionando o início da execução da instrução indicada por esse endereço e ainda, (I) = 0, realizando a transição para o estado NORMAL.

2.2 - PRIORIDADES E ROTINA DE DESCOBRIMENTO DE INTERRUPÇÕES

Quando ocorre uma interrupção, a primeira providência a tomar é descobrir qual a interface que provocou tal interrupção. Isto é realizado por um componente do NÚCLEO, denominado ROTINA DE DESCOBRIMENTO DE INTERRUPÇÕES, uma vez que essa informação não é imediatamente fornecida pelo "Hardware".

A ROTINA DE DESCOBRIMENTO DE INTERRUPÇÕES (RDI) faz uso da instrução "TESTA SE NÃO HÁ PEDIDO DE INTERRUPÇÃO DA INTERFACE" examinando um a um os sinais PDI. O diagrama da rotina pode ser visto na fig. 2.2.

Uma prioridade no atendimento das interrupções produzidas pelas interfaces é estabelecida de acordo com a ordem em que são realizados os testes dos sinais PDI pela rotina RDI. Assim, com relação à fig. 2.2., a interface de número j tem maior prioridade que a de número k, uma vez que um pedido de interrupção originado na interface k só será atendido quando não houver pedido interrupção da interface j.

"Atender" uma interrupção de interface j foi utilizado com o sentido de provocar um desvio incondicional para o programa que se inicia na posição IXj (fig. 2.2).

2.3 - INTERRUPÇÕES SÍNCRONAS E ASSÍNCRONAS

Uma interrupção provocada por uma instrução de máquina será chamada INTERRUPÇÃO SÍNCRONA ou PROGRAMADA. Este tipo de

DESCOBRIMENTO DAS INTERRUPÇÕES

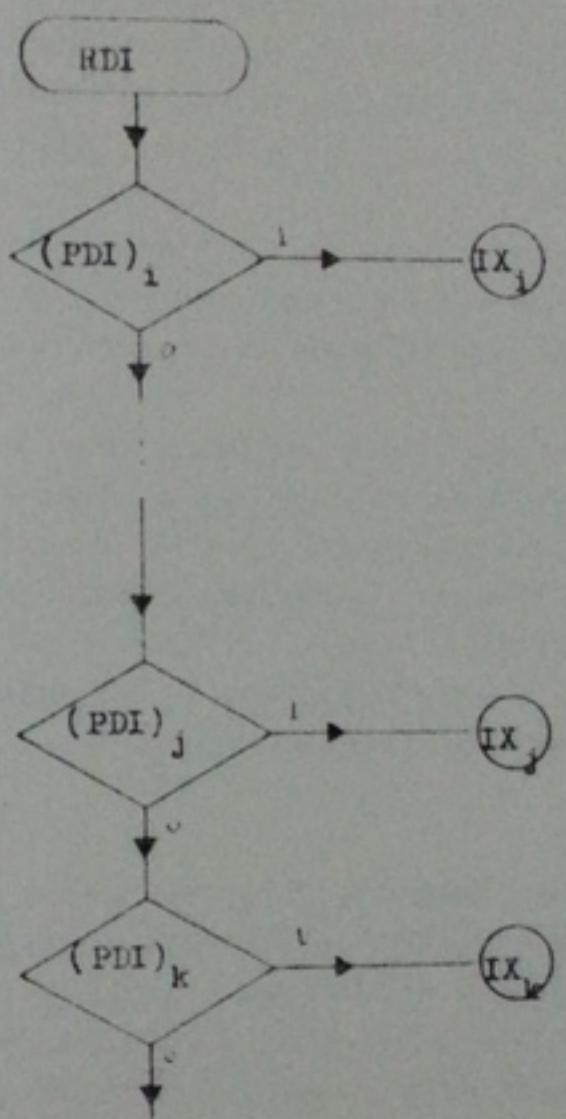



Fig. - 2.2

interrupção, dita às vezes "chamada de supervisor" (CSUP no computador G-10) ou "SVC" (Supervisor Call) é uma forma normalmente utilizada por um programa a fim de ter acesso ao núcleo ou ao Sistema Operacional. O "PATINHO FEIO" não dispõe atualmente de uma instrução especial para esse propósito. No caso, a solução escolhida para se simular uma instrução desse tipo foi utilizar uma interface especialmente para este fim.

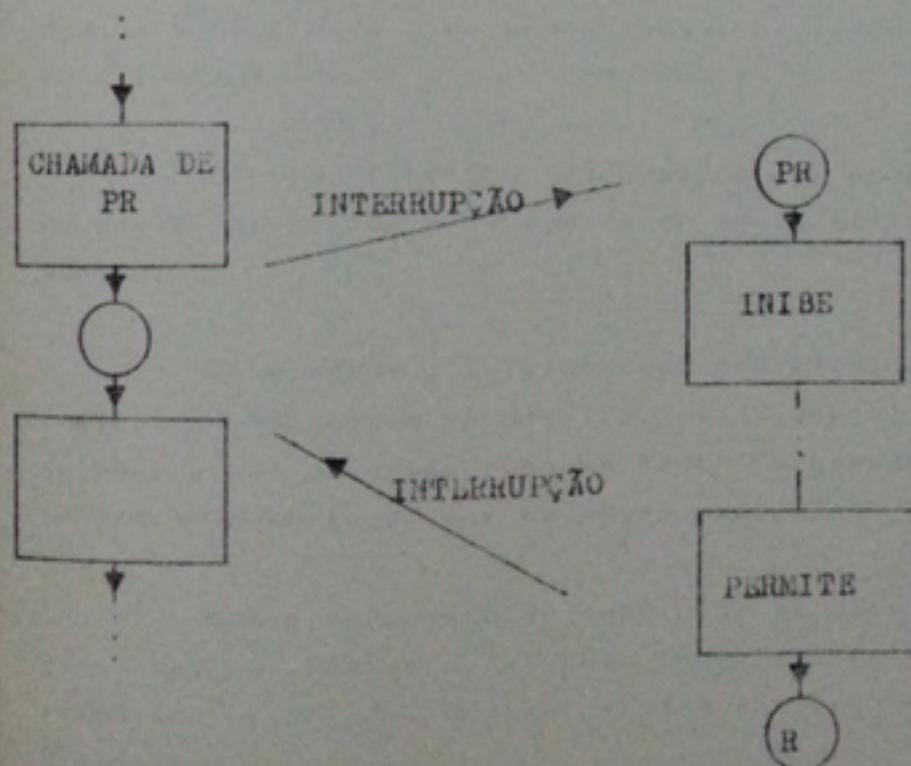
O único requisito para esta interface é que disponha dos circuitos padrões que implementam a lógica de interrupção conforme mencionada em 2.1. Esta será chamada interface S.

Supondo-se, por exemplo, que o sinal EST_S seja mantido permanentemente no nível 1, sempre que o sistema de interrupções estiver liberado, isto é, (PIT) = 1 e a U.C.P. em estado NORMAL, será suficiente executar a instrução "PERMITE INTERRUPÇÃO DA INTERFACE" ($(PRI)_S$ ← 1) para que ocorra a interrupção. A instrução "PERMITE INTERRUPÇÃO DA INTERFACE" quando utilizada em particular para a interface S, será chamada aqui de CSP (chamada de supervisor).

Para que a instrução CSP provoque efetivamente uma interrupção síncrona é necessário que a transição do estado NORMAL para INTERROMPIDO, ocorra antes do início da execução da instrução que segue o CSP. Os circuitos utilizados atualmente nas interfaces do "PATINHO FEIO" proporcionam a velocidade suficiente para que esta condição seja satisfeita

Uma outra condição é que a interface S tenha a mais alta prioridade estabelecida pela rotina RDI.

Com essa duas condições será possível simular uma instrução de chamada de supervisor.

Uma INTERRUPÇÃO ASSÍNCRONA será qualquer interrupção provocada por uma interface que não a interface S.

Estas interfaces, ao contrário desta última, estarão conectadas a dispositivos de entrada e saída, controlando suas

operações, de modo que estas interrupções poderão ocorrer em virtude de determinadas condições associadas a estes dispositivos durante sua operação, não sendo possível portanto prever-se exatamente em quais instantes estas irão ocorrer. Decorre deste deste fato, o caráter assíncrono deste tipo de interrupção.

2.4 - PRIMITIVOS E PROCESSOS

Conforme observado no início deste capítulo, o fato de uma instrução de máquina não ser interrompível, permite considerar todas as ações tomadas pela U.C.P. na execução da instrução como sendo um PRIMITIVO. Ainda, pelo que foi visto nos ítens anteriores, uma sequência de instruções realizadas pela U.C.P. no estado INTERROMPIDO, também poderá ser considerada um PRIMITIVO, uma vez que neste estado a U.C.P. não é interrompível.

Entretanto, é necessário considerar um aspecto importante na programação de primitivos. Por exemplo, considerando-se um trecho de programa ou uma rotina PR composta por uma série de instruções que se inicie pela instrução "INIBE INTERRUPÇÃO" ((PIT)←0) e termine pela instrução "PERMITE INTERRUPÇÃO" ((PIT)←1), essa subrotina não implementa um primitivo devido

à possibilidade de ocorrer uma interrupçao entre a execução de uma instrução de chamada de subrotina (PUG) e a instrução INIB e ainda entre a execução do PERM e o retorno de subrotina.

Uma forma de contornar o problema é obter o acesso às subrotinas que implementam os primitivos, através de uma interrupçao síncrona.Desta forma fica garantida a nao interruptibilidade desde a execução da CSP até a busca da instrução que a segue imediatamente. Esta última característica é proporcionada pela maneira de se realizar a transição do estado Interrompido para o estado NORMAL (2.1.).

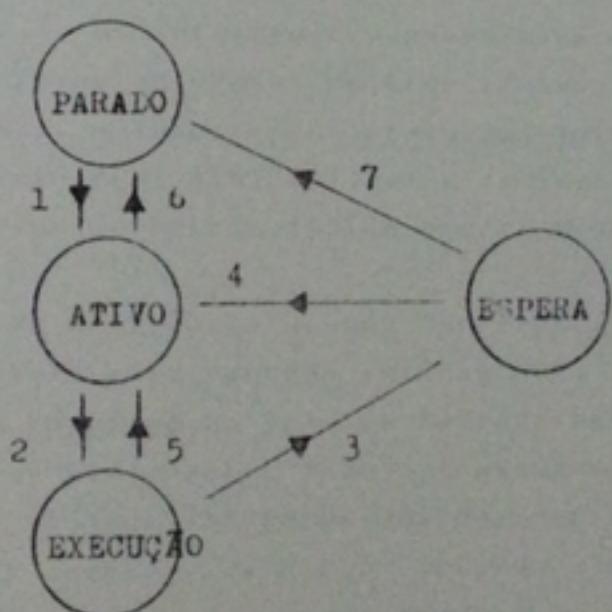
Grande parte do NÚCLEO será portanto composta por esse tipo de subrotina, acessíveis por meio da ROTINA DE DESCOBRIMENTO DE INTERRUPÇOES.

A RDI estabelece dessa forma, uma fronteira entre o NÚCLEO e os demais programas em execução e entre este e as interfaces de entrada e saída disponíveis no sistema.

O conjunto de primitivos e suas funçoes respectivas devem ser estabelecidas de acordo com os propósitos do sistema de programação visado. No presente trabalho, o objetivo principal dado a NÚCLEO será o de proporcionar os recursos necessários para o controle de um [illegible] número [illegible] processos independentes.

O termo PROCESSO irá designar um programa em execução na U.C.P. associado a um número de identificação disponível no NÚCLEO.

Um programa, inicialmente constituido por um conjunto de códigos em linguagem de máquina, residente na memória principal, só poderá ser encarado como um PROCESSO quando receber uma identificação autorizada por um outro processo.

Com o propósito de controlar as atividades de um ou mais processos existentes no sistema em um determinado instante, a estes sao associados quatro estados básicos. Tales estados serao discutidos a seguir.

2.5 - ESTADOS DE UM PROCESSO

Logo que um processo é identificado, este permanece no estado PARADO. Neste estado o processo não pode utilizar a U.C.P., ou seja, a U.C.P. não irá executar o programa associado a este processo sendo que o único recurso a este atribuido será a porção de memória ocupada pelo programa.

Somente a partir do estado ATIVO o processo terá condições de utilizar a U.C.P. para a execução de seus códigos. Toda vez que isto ocorrer o processo passa para o estado de EXECUÇÃO.

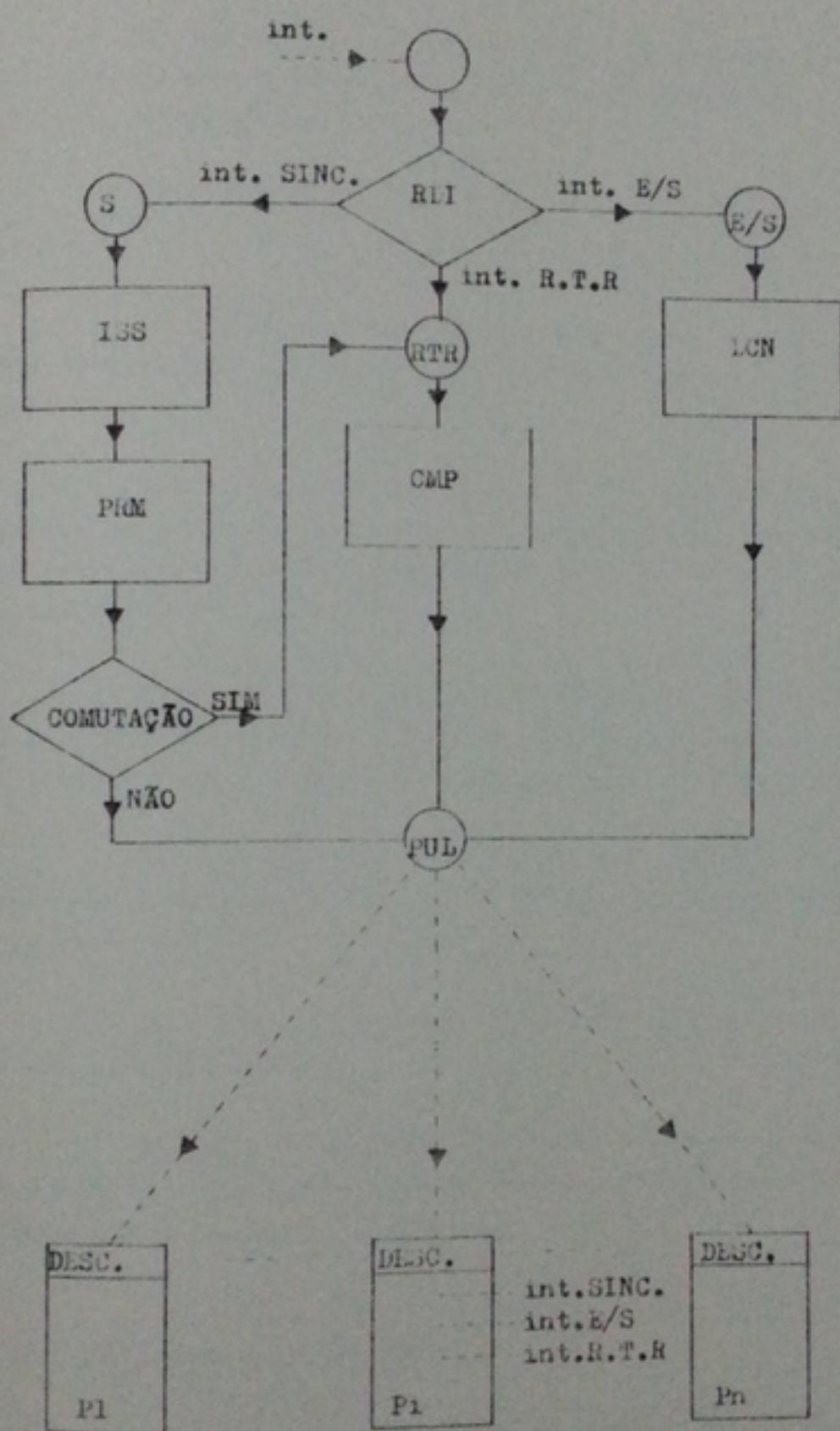
Nos intervalos de tempo em que por qualquer motivo a execução do processo está suspensa, este assume o estado de ESPERA.

A figura abaixo ilustra estes dados bem como as transições possíveis entre eles.

A escolha deste estados, bem como as transições que podem ser realizadas entre eles, decorrem do fato de se admitir a existência de vários processos simultâneos competindo com os limitados recursos disponíveis no sistema, bem como das interações que possam existir entre eles.

Supôs-se que em cada instante, exceto aquele que utiliza a U.C.P. todos os processos podem ser agrupados entre aqueles que só dependem da liberaçao da U.C.P. para que possam utilizá-la (ATIVOS), os que dependem da ocorrência de algum evento específico produzidos por outros processos para que possam tornar-se ativos, e por último, processos recém criados ou prestes a serem removidos do sistema e que nao participam diretamente das atividades desenvolvidas neste instante.

Essa organizaçao é típica dos sitema de multiprocessos (ref. (10) comportando no entanto subdivisoes nos estados principais em subestados ou ainda a inclusao ou nao de determinadas transiçoes de acordo com os algorítmos utilizados no controle dos processos.

Será admitida inicialmente a organizaçao esquematizada na figura 2.3 , cuja finalidade será esclarecida nos ítens seguintes.

Um esquema da organizaçao geral do NÚCLEO é apresentada na fig. 2.3. As interrupçoes provocadas pelas interfaces ligadas à U.C.P. foram agrupadas em tres classes, a interrupçao síncrona (INT. SINC), a interrupçao provocada interface ligada a um Relógio de Tempo Real (INT.R.T.R.) e as interrupçoes devidas aos dispositivos de Entrada/Saída propriamente ditos (INT.E/S).

Na ocorrência de uma interrupçao, a rotina RDI transfere o controle para uma das rotinas DCN, ISS ou CMP conforme a origem da interrupçao seja de Entrada/Saída, síncrona ou do R.T.R. respectivamente. Todo o tratamento de uma interrupçao é realizado no modo INTERROMPIDO, de maneira que no final deste tratamento, o controle da U.C.P. retorna a um dos processos em andamento e que esteja no estado ATIVO.

A cada processo esta associado um DESCRITOR,constituido por 7 palavras, nas quais é armazenado o "STATUS" do processo quando este é interrompido e de onde é lido quando o processo for retomado.

Através deste procedimento é possível retomar qualquer um dos processos em andamento após o tratamento de qualquer inter

ORGANIZAÇÃO DO NÚCLEO

Fig. - 2.3

rupçao.

A rotina DCN controla as operaçoes dos dispositivos de Entrada/Saída realizando a transferência dos dados entre a memó ria e os dispositivos.

Após o atendimento de uma interrupçao deste tipo, o con trole da U.C.P. retorna ao último processo que foi interrompido.

As rotinas ISS e PRM sao responsáveis pela análise de uma interrupçao síncrona e execuçao do primitivo designada por esta.

Após a execuçao do primitivo, o controle da U.C.P. é en tregue ao processo que promoveu a execuçao deste primitivo ou a um outro processo no estado ATIVO. Esta escolha depende do parti cular primitivo executado.

A rotina CMP realiza a COMUTAÇAO de processos, isto é, a passagem do controle da U.C.P. de um processo para outro. Esta comutaçao é realizada sempre que ocorrer uma interrupçao R.T.R.

As interrupçoes do R.T.R. sao periódicas, o que permite que o uso da U.C.P. seja distribuido mais uniformente entre os processos ativos.

Este esquema chamado "ROUND-ROBIN" foi utilizado devido a sua simplicidade.

Uma descriçao mais detalhada das rotinas mencionadas aci ma, bem como das estruturas de dados utlizadas por estas, será fornecida a seguir.

2.6 - IMPLEMENTAÇAO DO NÚCLEO

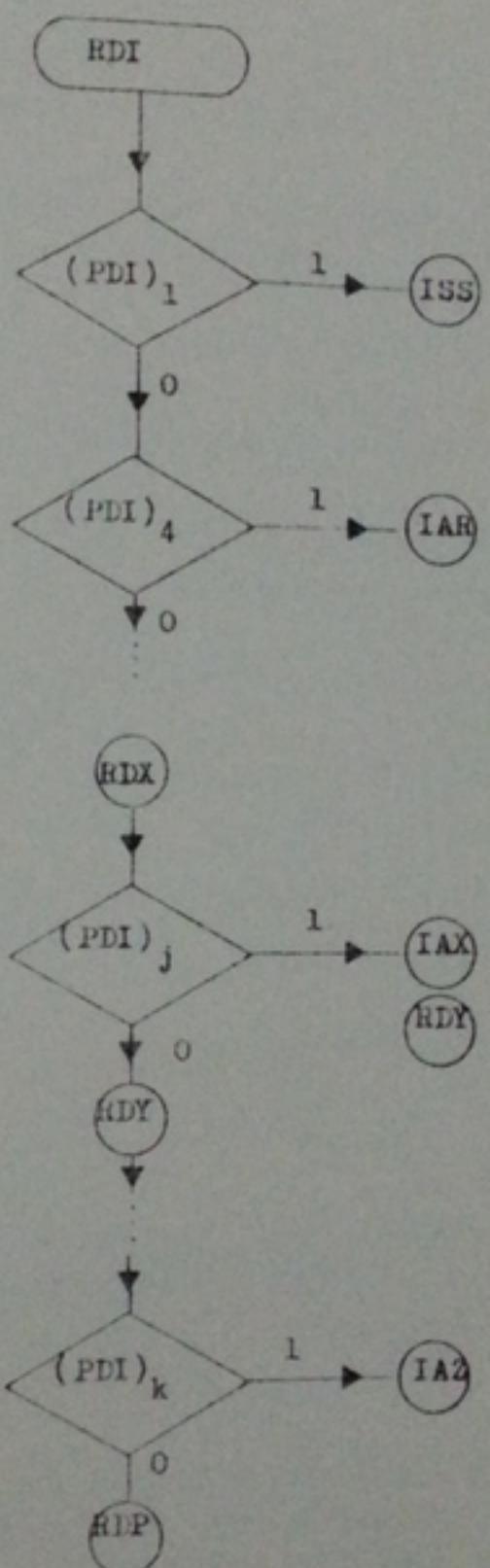
Na implementaçao do NÚCLEO, foram utilizados os progra mas MONTADOR RELOCÁVEL (2) e o CARREGADOR RELOCÁVEL (15) dispo níveis no "PATINHO FEIO" na montagem das subrotinas que o com poem, descritas na linguagem do MONTADOR RELOCÁVEL.

2.6.1 - ROTINA DE DESCOBRIMENTO DE INTERRUPÇOES (RDI)

Conforme mencionado anteriormente, a RDI estabelece o ponto de acesso ao NÚCLEO do sistema.

A posição de rótulo RDI (fig. 2.4) é alcançada por meio de interrupçoes, devendo ocupar portanto as posiçoes de endereços 4 e 5 da memória.

As interfaces de números 1 e 4 foram atribuídas as funçoes de produzir as interrupçoes síncronas e as interrupçoes do RELÓGIO DE TEMPO REAL (2.6.4.) respectivamente, sendo consideradas as de maior prioridade pelo NÚCLEO. O tratamento dessas interrupçoes é realizado pelas rotinas ISS e ISR.

As demais interfaces, até no máximo 11, estarao ligadas aos dispositivos de ENTRADA e SAÍDA.

No teste de uma determinada interface j, foi tomada a providência de se rotular cada teste (RDX) a fim de facilitar em primeiro lugar, ignorar o resultado de testes (1) quando a interface j nao está conectada substituindo-se IAX por RDY, ou seja passando-se para o teste da próxima interface. Em segundo lugar a troca de prioridades entre duas interfaces por j e k poder ser realizada trocando-se IAX por IAZ, a instruçao (PDI) j por (PDI)k e vice-versa.

As interrupçoes produzidas pelo painel (BOTAO DE INTERRUPÇAO DO PAINEL (I)) terá a menor prioridade, sendo tratada na rotina de rótulo R.D.P. Nao existe nenhum teste específico para este último tipo de interrupçao, de modo que a sua descoberta é feita por exclusao.

2.6.2 - ROTINA DE TRATAMENTO DE INTERRUPÇOES SÍNCRONAS (ISS)

A rotina ISS compoe-se de um conjunto de subrotinas que implementam um determinado conjunto de primitivo.

ROTINA DE DESCOBRIMENTO DE INTERRUPÇÕES

Fig. - 2.4

Para isto, dispoe de uma estrutura de dados própria e ainda uma coleção de subrotinas auxiliares.

2.6.2.1 - GERAÇAO DA INTERRUPÇAO SÍNCRONA

Conforme estabelecendo anteriormente, uma interrupçao síncrona é realizada pela instruçao CSP.

Esta comporta um determinado número de parâmetros destinados a identificar a interrupçao e transmitir informaçoes necessárias às subrotinas que implementam os primitivos.

Na forma geral tem-se uma instruçao do tipo CSP (n_1, n_2 ... n_k) onde n_1, n_2, ...n_k sao os parâmetros a serem transmitidos.

Na linguagem do MONTADOR (2), esta será implementada por uma "sequência de chamada" do tipo:

```
FUNC    /15    C115
DEFC    n1     n1
DEFC    n2     n2
 .              .
 .              .
 .              .
DEFC    nk     nk
```

Ao lado dos mnemônicos, estao os códigos de máquina (notaçao hexadecimal) da "sequência de chamada" produzida pelo MONTADOR ou por outro compilador de uma linguagem que disponha deste tipo de comando.

O primeiro parâmetro, n_1, identifica qual o primitivo que deve ser executado, enquanto que os demias, se houver, serao parâmetros do particular primitivo solicitado.

É necessário notar que após a execuçao da instruçao FNC/15 (funçao de entrada e saída para a interface 1, permitindo interrupçao), o CONTADOR DE INSTRUÇOES aponta a posiçao do parâmetro n_1, e é colocado nas posições 2 e 3 da memória.

2.6.2.2 - ESTRUTURA DE DADOS DA ROTINA ISS

A rotina ISS contém um conjunto de dados estruturados, com o propósito de descrever a situação de cada processo em andamento no sistema.

A cada processo está associado um DESCRITOR, que é constituído por uma sequência de 7 palavras consecutivas contendo o "STATUS" do processo e outras informações gerais a respeito do processo.

O "STATUS" de um processo é igual ao "STATUS" da U.C.P. em cada instante , quando este estiver em EXECUÇÃO. Compreende os valores do ACUMULADOR, EXTENSAO, INDICE, CONTADOR DE INSTRUÇOES (CI), e os sinais de TRANSBORDO (T), e "VAI-UM" (V), conforme a referência (1).

Quando ocorrer uma comutação de processos, o "STATUS"da U.C.P. é armazenado no DESCRITOR do processo que sofreu a interrupção, enquanto que esta irá assumir o "STATUS" contido no DESCRITOR do novo processo a ser executado.

Estes DESCRITORES (fig. 2.5) estão localizados nas 7 posiçoes consecutivas que antecedem a primeira palavra do programa associado ao processo e contêm :

ACC : ACUMULADOR

EXT : EXTENSAO

IND : ÍNDICE

$TVCI_1$: SINAIS T (bit 7), V (bit 6) e CONTADOR DE INSTRUÇOES (bits 0 a 3).

CI_0 : CONTADOR DE INSTRUÇOES (bits menos significativos)

XD_1 : DIMENSAO EM NÚMERO DE PALAVRAS DO PROGRAMA (mais significativos)

D_0 : DIMENSAO EM NÚMERO E PALAVRAS DO PROGRAMA (menos significativos)

TABELA DE DESCRIÇÃO DOS PROCESSO (TDP)

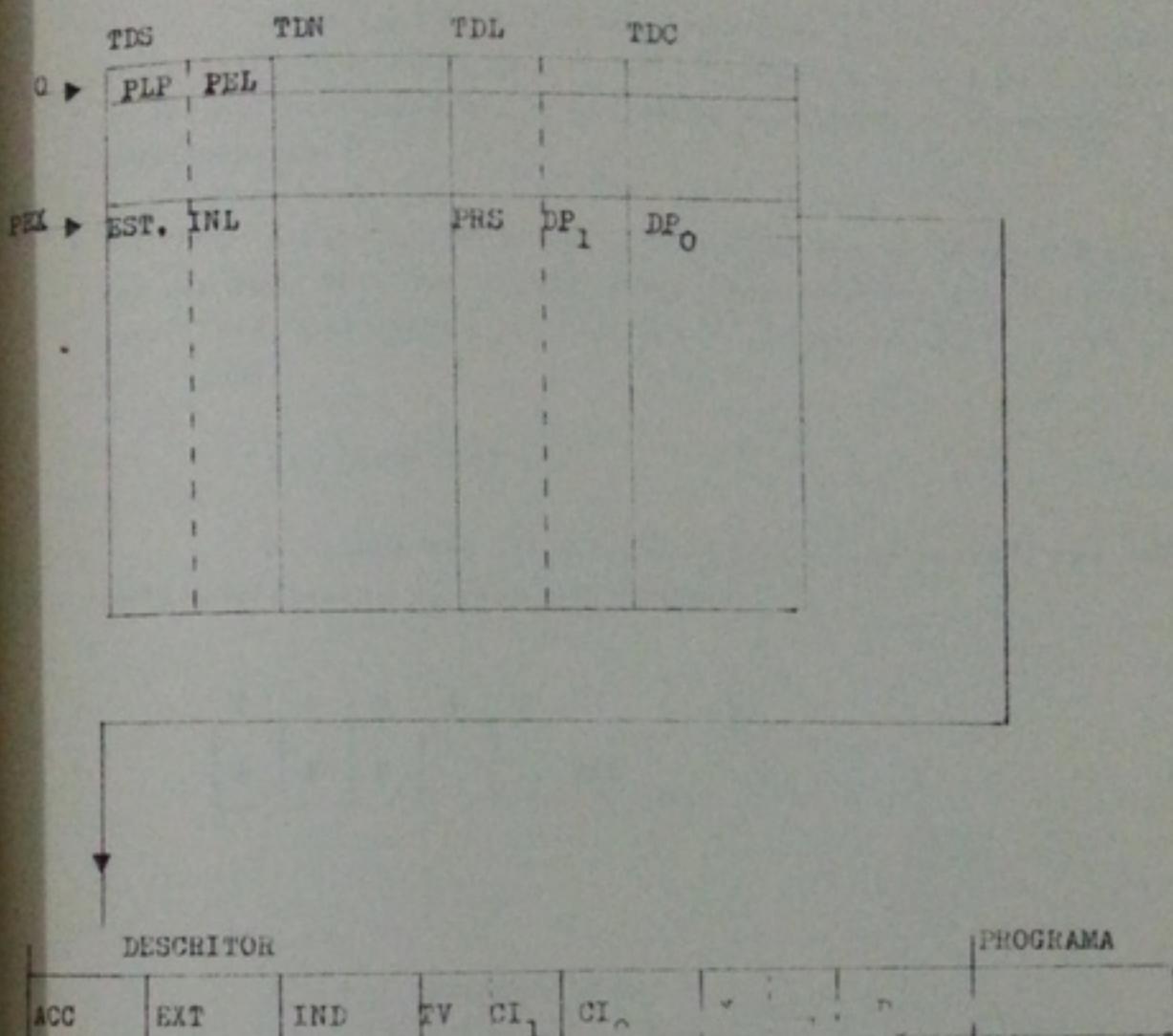



Fig. - 2.5

Os 4 bits mais significativos de XD_1 podem ser usados para um outro propósito qualquer.

A cada processo em andamento está associado um número de identificaçao entre 1 e 15. Esta identificação é utilizada como índice correspondente a uma linha da TABELA DE DESCRIÇAO DOS PROCESSOS (TDP).

A tabela TDP (fig. 2.5) contém quatro colunas denominadas por TDS, TDN, TDL e TDC, sendo cada elemento destas colunas constituido por uma palavra (8 bits), em número de 15 elementos por coluna.

a) COLUNA TDS

A coluna TDS contém informações relacionadas com o ESTADO, codificadas da seguinte forma :

7	6	5	4	3 ... 0
A	E	P		INL

A = 1 : PROCESSO ATIVO

E = 1 . PROCESSO EM ESPERA

P = 1 : PROCESSO PARADO

O número máximo de processos simultâneos (15) foi escolhido tendo em vista unicamente a extensao de memória ocupada pela tabela TDP em relaçao ao total de posiçoes de memória disponível (4K) e por facilidade de manipulaçao desta.

A possibilidade de adiçao e remoçao dinâmica de processos, o que irá provocar a existência de linhas utilizadas e linhas vazias na TDP, proporcionou a necessidade da implementaçao de um esquema de inserçao e retirada de elementos desta tabela.

Este esquema está baseado em duas listas encadeadas,uma percorrendo as linhas ocupadas e a outra as linhas vazias de TDP.

Para este efeito, os quatro bits menos significativos de cada elemento da coluna TDS, (INL) são utilizados como INDICADOR DA LISTA, possibilitando referenciar-se qualquer linha da tabela.

Estas listas encadeadas são implementadas utilizando-se as variáveis PLP e PEL indicando a primeira linha da lista de processos e a primeira linha da lista vazia respectivamente e o encadeamento continuado pelos valores de INL. Em particular INL = 0 determina o final de uma lista.

De um modo geral, sempre que uma lista contiver encadeamento, este é realizado utilizando-se a primeira linha da tabela com os quatro bits mais significativos (PLP) indicando o primeiro elemento utilizado e os quatro bits menos significativos (PEL), o primeiro elemento livre.

Com esta padronizaçao, uma mesma subrotina (PQL) poderá realizar as operações de inserçao e retirada de elementos de uma tabela encadeada bastando que se forneça o endereço do início da tabela.

Uma situação particular destas listas está esquematizada na fig. 2.6 .

b) COLUNA TDN

A cada processo associa-se um CÓDIGO (1 a 255) indicado por um elemento da coluna TDN.

Este CÓDIGO constitui-se na maneira pela qual um processo pode referenciar um outro a longo prazo, uma vez que a identificaçao (índice na TDP) do processo pode sofrer alterações dinamicamente.

c) COLUNAS TDL E TDC

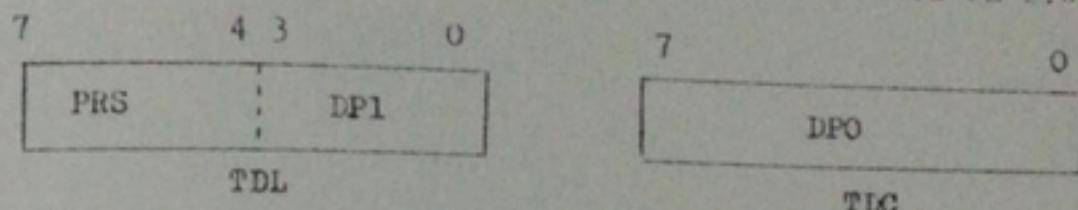
Os elementos das colunas TDL e TDC contém um apontador para o DESCRITOR do processo correspondente. Esse apontador (12

LISTAS DA TABELA DE DESCRIÇÃO DOS PROCESSOS

Fig. - 2.6

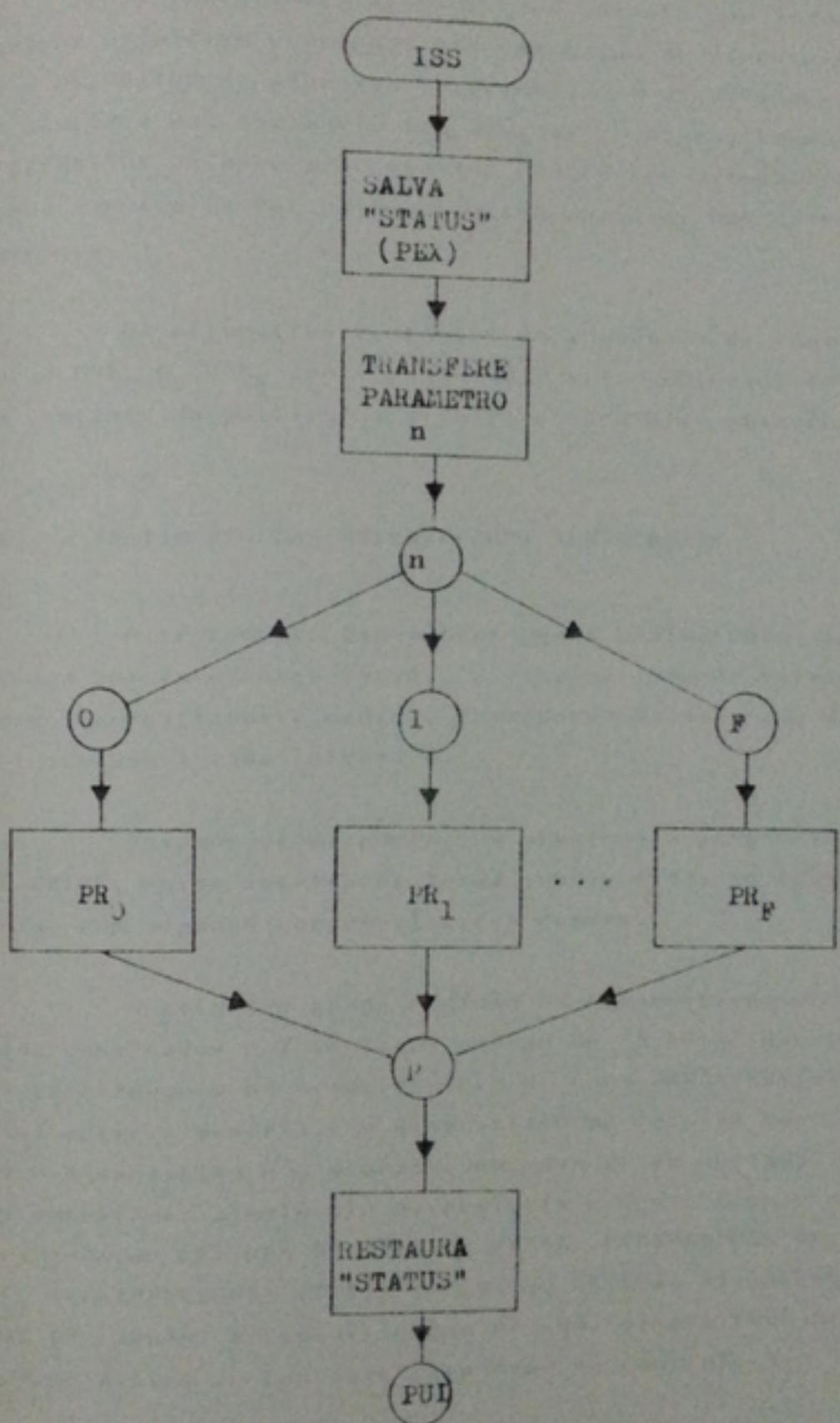
bits) é codificado em TDC e pelos 4 bits menos significativos de TDL .

Os quatro bits mais significativos de TDL são utilizados como complementação de uma tabela TDH que será descrita em 2.6.7.

PRS : Indicador da lista de processos subordinados

DP_1, DP_0 : Apontador para o DESCRITOR do processo.

2.6.2.3 - ESTRUTURA DA ROTINA ISS

A fig. 2.7 ilustra o diagrama da rotina ISS.

Inicialmente, após a execuçao da instruçao CSP realizada pelo processo em EXECUÇAO, identificado pela variável PEX , o "STATUS" desse processo é salvo em algumas variáveis temporárias.

O endereço dos parâmetr s (EPI, 3) é obtido a partir dos conteudos das posiçoes 2 e 3 da memória, após o que o primeiro parâmetro (n_1) é obtido. O valor n_1 identifica o primitivo a ser utilizado e caso esse valor exceda um valor máximo (NPP) o primitivo erro (NPR) é assumido.

Após a identificaçao do primitivo através de n_1, é realizado um desvio para a subrotina que executa esse particular primitivo PR_0 a PR_n, onde n = NPP.

A execuçao de cada primitivo termina com a restauraçao do "STATUS" da U.C.P. por meio das variáveis temporárias, a liberaçao dos pedidos de interrupçao da interface S, fazendo-se $(PDI)_S = 0$, e finalmente por meio da instruçao PUL, a U.C.P. realiza a transiçao do estado interrompido para o estado normal.

ESTRUTURA DA ROTINA ISS

FIG. - 2.7

Os DESCRITORES de processos somente são atualizados quando algum primitivo ocasionar uma comutação de processos. Neste caso, o DESCRITOR do processo interrompido é atualizado com os valores contidos nas variáveis temporárias e estas assumem os valores do DESCRITOR do novo processo que deverá ser acionado. Essas operações são realizadas pela subrotina auxiliar CMP de comutação de processos.

Os parâmetros restantes da sequência de chamada da instrução CSP, n_2 a n_k são transferidos pela subrotina TPI e estão sob controle do particular primitivo escolhido através de n_1.

2.6.3 - TRATAMENTO DAS INTERRUPÇÕES ASSÍNCRONAS

O tratamento das interrupções assíncronas, em geral provocadas por interfaces ligadas a dispositivos de entrada e saída, não é necessariamente padrão, dependendo da natureza do dispositivo conectado à cada interface.

Visando os dispositivos atualmente disponíveis no "PATINHO FEIO", estas interfaces foram subdivididas em três grupos de acordo com algumas características comuns.

O primeiro grupo engloba os dispositivos de entrada e saída conectados a interfaces padrão de "8 bits" que podem provocar interrupções na transferência de cada CARÁCTER(palavra de 8 bits) entre a memória e o dispositivo no caso de uma saída ou entre o dispositivo e a memória, no caso de um entrada. A maioria das interfaces atualmente disponíveis é deste tipo e controla as operações de LEITORA DE FITA DE PAPEL, PERFURADORA DE FITA DE PAPEL, TELEIMPRESSORA (TTY), inclusive LEITORA DE CARTÕES e IMPRESSORA DE LINHAS. A especificação de cada dispositivo utilizado no Sistema Básico de Controle será dado no capítulo 3.

Um segundo grupo, reuniria as interfaces ligadas a dispositivos especiais como por exemplo DISCOS MAGNÉTICOS, FITAS MAGNÉTICAS nos quais a transferência de grandes quantidades de dados

em velocidades mais altas que os anteriores exigem interfaces capazes de produzir acessos à memória principal sem a intervenção direta da U.C.P., proporcionando um tratamento distinto das interrupções, para cada caso. Neste grupo foram incluídos ainda eventuais interfaces ligadas a CONVERSORES tipo ANALOGO/DIGITAL ou DIGITAL/ANALÓGICO ou ainda MULTIPLEXADORES deste tipo de dispositivo.

Finalmente, o terceiro grupo irá reunir as interfaces que interagem, por meio de interrupções, diretamente com o NÚCLEO como é o caso da "interface S" (2.3) e uma interface chamada RELÓGIO DE TEMPO REAL capaz de produzir interrupções periódicas. No caso, somente esta última é considerada geradora de interrupções assíncronas conforme definido em 2.3 . Neste trabalho apenas para os dispositivos do primeiro grupo serão implementados rotinas de tratamento, esquematizadas em 2.6.2.3 .

2.6.4 - ACIONAMENTO DE PROCESSOS

O acionamento de um processo consiste em colocá-lo no estado de "EXECUÇÃO", ou em outras palavras fornecer a U.C.P. para o uso exclusivo do processo durante um determinado intervalo de tempo.

Durante a execução de um processo, tendo este controle exclusivo sobre a U.C.P., a oportunidade de se acionar um outro processo somente ocorre quando este for interrompido.

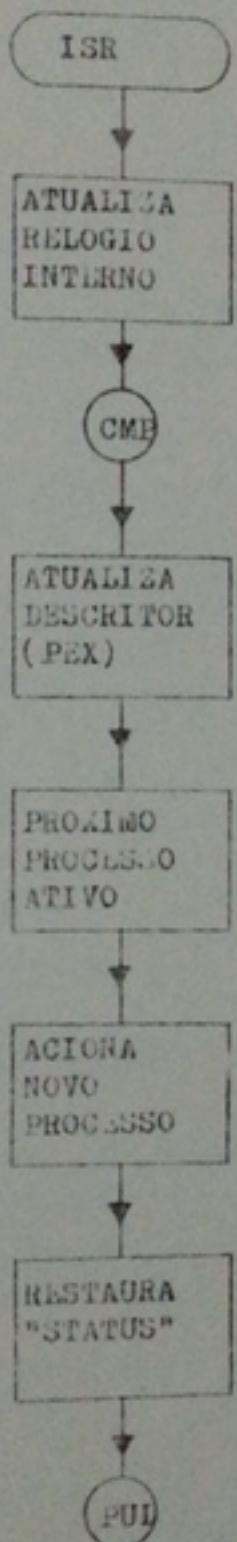
Esta interrupção pode ser provocada tanto por iniciativa do processo em execução, através de uma interrupção síncrona ou por eventos externos como é o caso de uma interrupção assíncrona. A primeira hipótese é imprevisível, podendo ocorrer casos em que um processo utilize a U.C.P. por tempos indefinidos, por exemplo, por um mal funcionamento do processo.

Com o propósito de garantir uma distribuição uniforme da U.C.P. entre os processos em andamento no sistema, foi utilizado o esquema pelo qual cada processo recebe periodicamente um intervalo de tempo fixo (time slice) durante o qual este dispõe da U.C.P.

Este esquema, também chamado "ROUND-ROBIN" (3) é implementado utilizando-se o recurso da interface RELÓGIO DE TEMPO REAL.

O RELÓGIO DE TEMPO REAL, conectado na interface de número 4 e com o segundo nível de prioridade (2.2), é capaz de produzir interrupções periódicas. O perído pode ser programado por meio de instruções especiais para a interface, podendo ser fixado entre 1 milisegundo a 100 segundos.

As interrupções assíncronas produzidas pelo RELÓGIO DE TEMPO REAL (RTR) são atendidas pela rotina ISR. Esta rotina, esquematizada na fig. 2.8, após cada interrupção, atualiza o descritor do processo interrompido, da mesma forma que realizada pela rotina ISS e passa a obter o próximo processo ativo.

A busca do próximo processo ativo é realizada explorando-se o encadeamento de todos os processos em andamento, disponível na TABELA DE DESCRIÇÃO DOS PROCESSOS (TDP).

A partir da posição indicada pela variável PEX, correspondente ao processo interrompido, a lista da coluna TDS é percorrida e o primeiro processo encontrado no estado ATIVO é escolhido.

A variável PEX é atualizada conforme o valor obtido acima, colocando o processo correspondente no estado de "EXECUÇÃO". Essas operações são realizadas pela subrotina "PROXIMO PROCESSO ATIVO" (PAT).

Obtido o novo valor de PEX, o pedido de interrupção do R.T.R. é atualizado ($(PDI)_4 \cdot 0$), uma nova interrupção é permitida e por fim o novo processo é acionado da mesma forma que na rotina ISS.

A fixação do valor do intervalo de tempo atribuída a cada processo não é imediata, sendo necessário observar-se o comportamento geral do sistema para cada valor experimental a fim de se obter um valor ótimo para cada conjunto de processos em operação simultânea.

ROTINA DE TRATAMENTO DAS INTERRUPÇÕES DO RELÓGIO DE TEMPO REAL

Entretanto um valor mínimo pode ser avaliado, considerando-se o tempo utilizado pela rotina ISR. Para essa implementação, este tempo foi avaliado em cerca de 2ms, de modo que o intervalo de tempo não deve ser menor que este. Um primitivo especial (PRT) permite que o "Time Slice" seja modificado de acordo com as necessidades.

2.6.4.1 - PRIMITIVOS RELACIONADOS COM O RELÓGIO DE TEMPO REAL

Três primitivos foram implementados a fim de possibilitar o controle do RELÓGIO DE TEMPO REAL e da variável RELÓGIO INTERNO.

a) PROGRAMA RELÓGIO DE TEMPO REAL (PRT)

Mnemonicamente, PRT(n) programa o RELÓGIO para pedir interrupções periódicas, com perídos de 10^n ms, $0 \leq n \leq 5$. A codificação desse primitivo é a seguinte:

CSP

0

n

onde 0 identifica esse primitivo.

b) LÊ RELÓGIO INTERNO (LRI)

A variável RELÓGIO INTERNO (RIT) constituida por três palavras consecutivas pode ser lida pelo primitivo LRI (L_0, L_1, L_2) onde L_0, L_1, L_2 são três posições que seguem a instrução CSP, onde serão armazenados o valor de RIT.

CSP

1

L_0

L_1

L_2

A variável RIT é incrementada a cada interrupção do R.T.R.

c) ALTERA RELÓGIO INTERNO (ARI)

Analogamente a LRI, o primitivo ARI (L_0, L_1, L_2) transfere os valores consecutivos L_0, L_1, L_2 para as posições reservadas para a variável RIT.

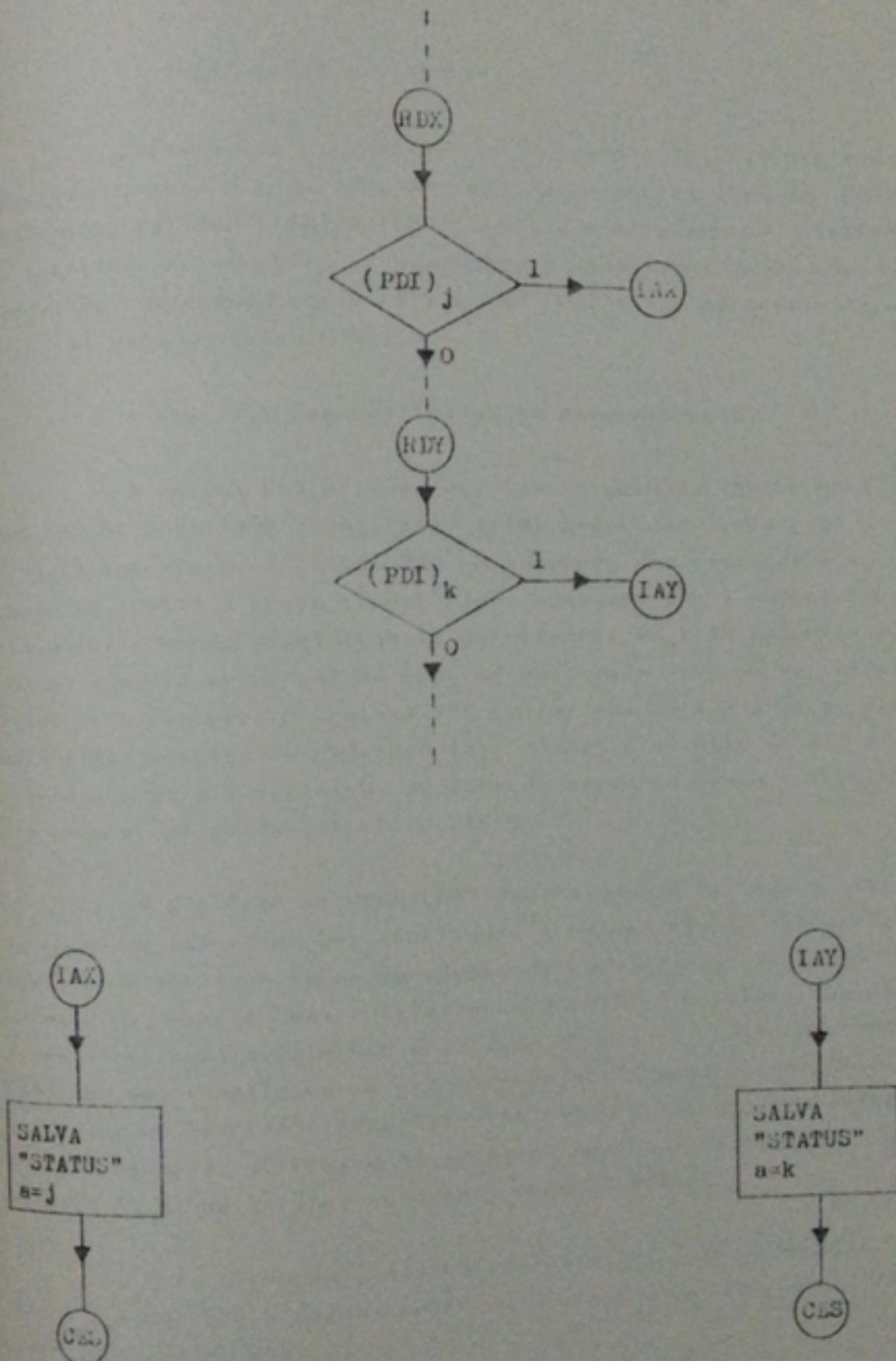
CSP

2

L_0

L_1

L_2

2.6.5 - CANAL CONCENTRADOR DE ENTRADA E SAÍDA

O têrmo CANAL CONCENTRADOR DE ENTRADA E SAÍDA foi utilizado para referenciar o conjunto de subrotinas e estruturas de dados do NÚCLEO destinados ao tratamento das interrupções e controle das interfaces padrão de 8 bits, conforme mencionadas em 2.6.3. Uma analogia com os CANAIS CONCENTRADORES DE ENTRADA E SAÍDA implementados em "HARDWARE" (11) é aqui realizada, tendo-se por objetivo a simulação pelo NÚCLEO de um processader de entrada e saída para os dispositivos mencionados no primeiro grupo em 2.6.3 .

As interrupções produzidas por uma interface ligada ao canal concentrador são identificadas pela rotina RDI, conforme descrito anteriormente, e realizada a identificação um desvio incondicional é feito para uma subrotina de atendimento (IAX) com endereços distintos para cada interface. Essas subrotinas são semelhantes e realizam as seguintes operações :

a) SALVAR O "STATUS" DA U.C.P.

O "STATUS" da U.C.P. por ocasião da interrupção, exceto o valor do CONTADOR DE INSTRUÇÕES que é colocado automaticamente nas posições 2 e 3, é salvo em uma área comum do canal mediante a chamada da subrotina SST .

RDX
(PDI)j
1
IAX
0
RDY
(PDI)k
1
IAY
0

IAX
SALVA
"STATUS"
a=j
CES

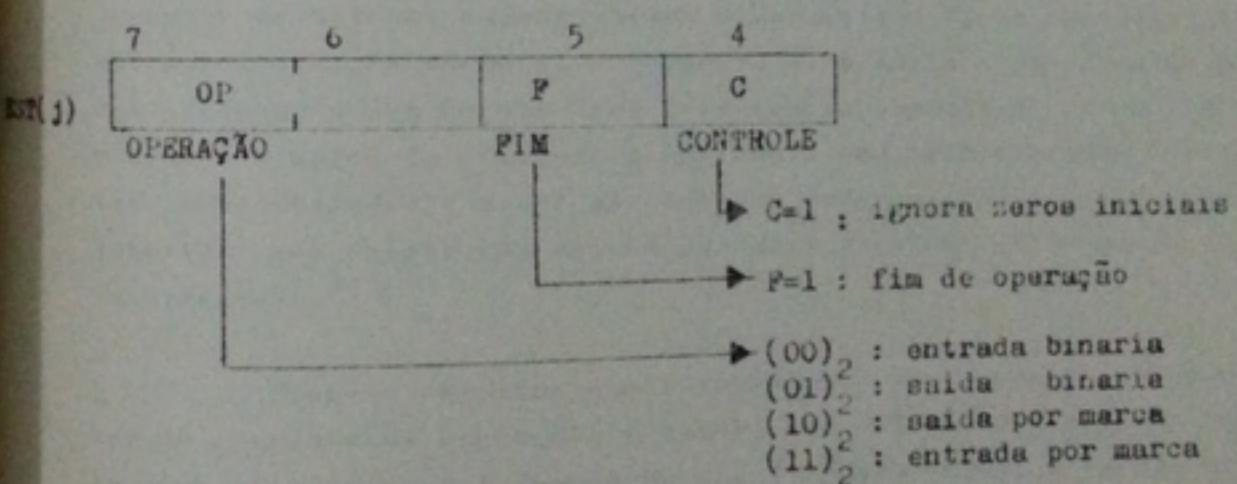
IAY
SALVA
"STATUS"
a=k
CES

FIG. - 2.9

b) IDENTIFICAR O DISPOSITIVO

Uma vez que o controle das operações dos dispositivos é semelhante, este é realizado por uma única rotina chamada CONTINUADORA DE ENTRADA E SAÍDA (CES). Após o salvamento do "STATUS" cada interface se identifica carregando o acumulador com o número da interface correspondente e em seguida realizando um desvio incondicional para a rotina CES .

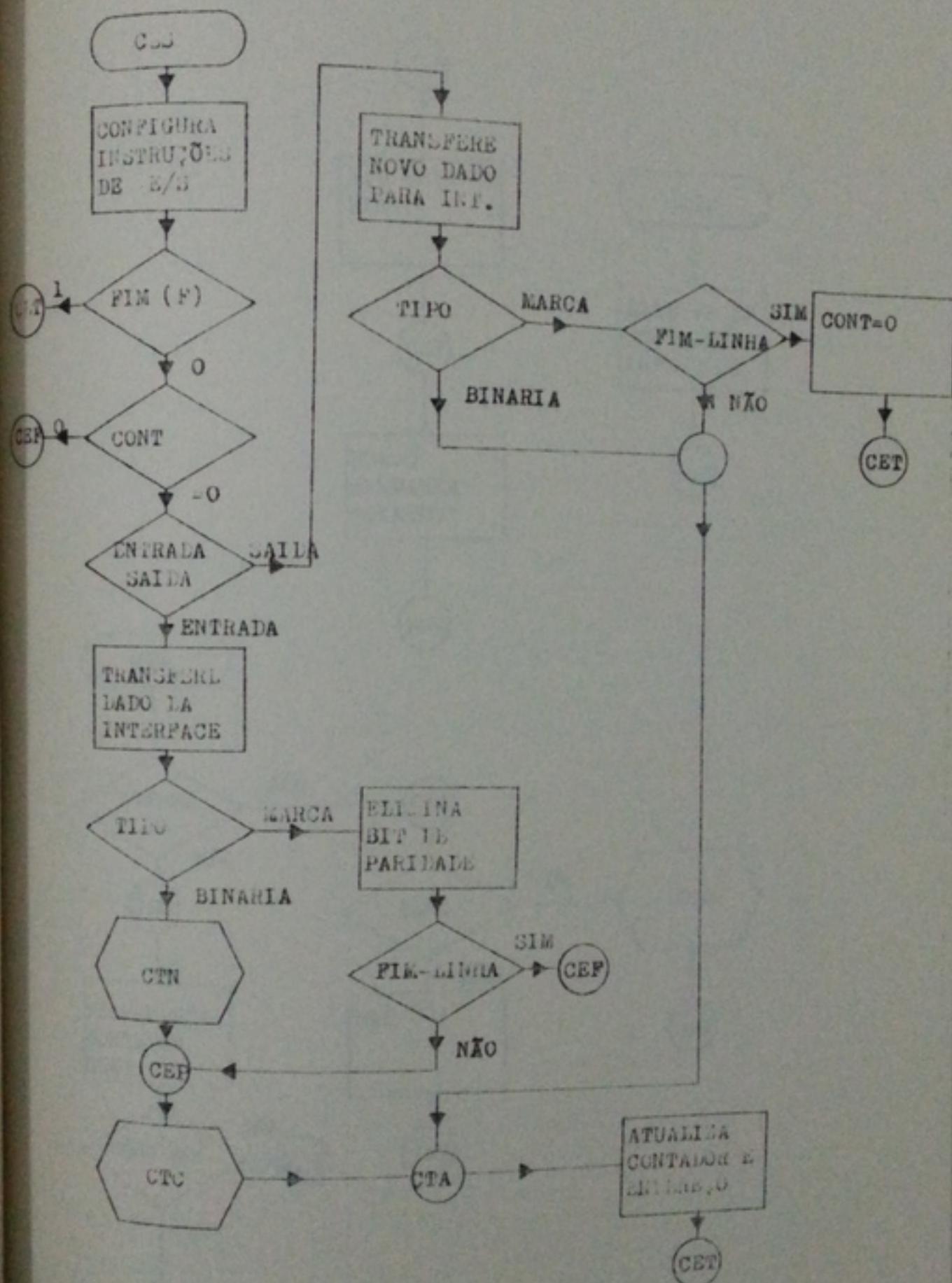
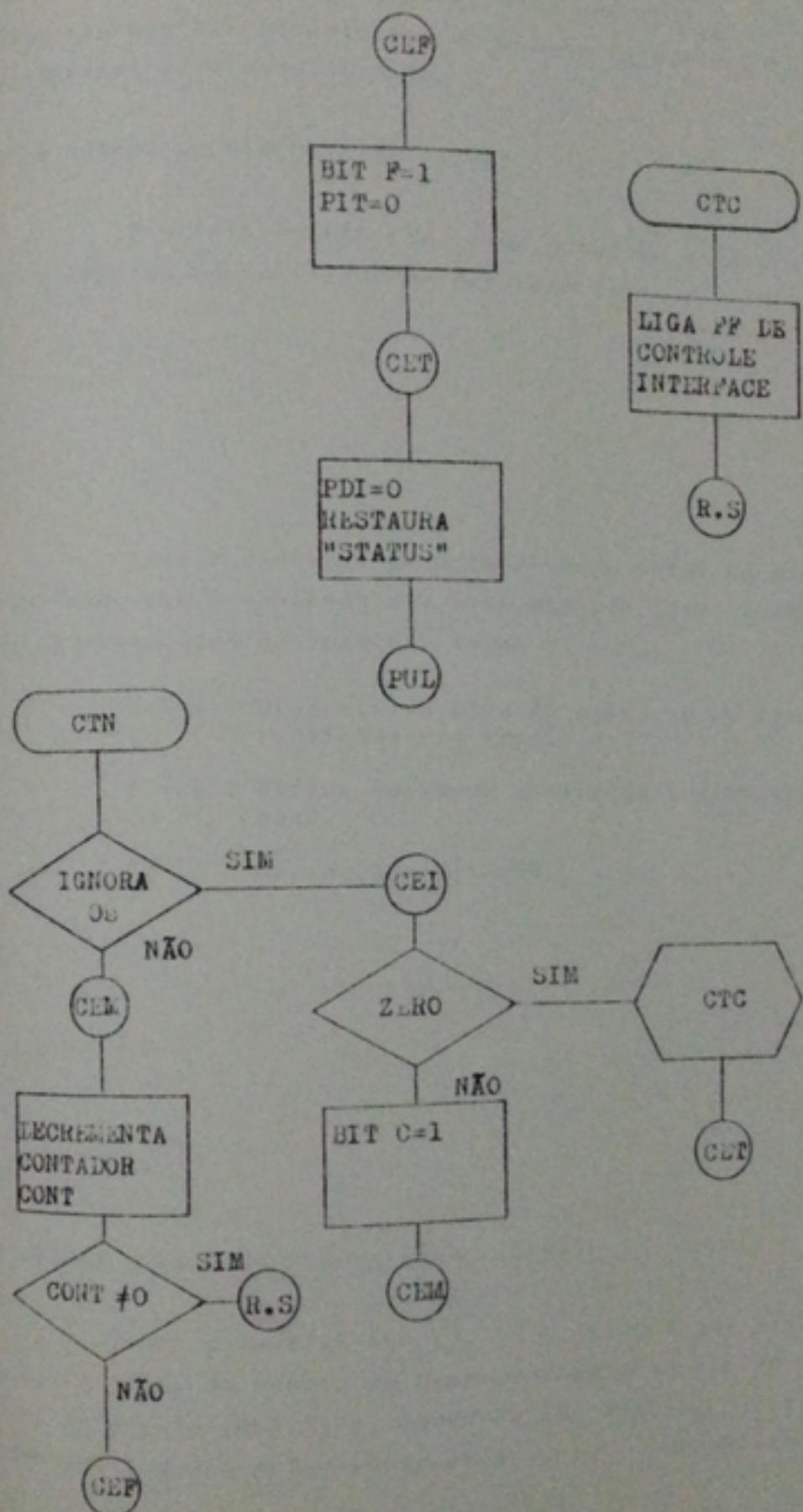
A fig. 2.9 esquematiza esses procedimentos.

A rotina CES utiliza uma tabela chamada TABELA DE ESTADOS DOS DISPOSITIVOS (TED),(fig. 2.10) onde cada linha está associada à interface. A coluna TEC corresponde aos contadores de caracteres (CONT) a serem transferidos enquanto que a coluna TEB contém a parte menos significativa do endereço (E_0) da primeira posição na memória principal de onde ou para onde deverão ser transferidos os caracteres. A coluna TEE contém nos bits 0 a 3 a parte mais significativa do endereço (E_1) citado e os bits de 4 a 7 são reservados para a descrição do tipo de operação sendo realizada bem como o estado da interface (EST).

O bit "FIM DE OPERAÇÃO" indica quando "1" que a última operação da interface foi realizada. A existência de dispositivos capazes de realizar tanto operações de entrada como de saída de caracteres, caso de uma teleimpressora (TTY) é prevista assinalan do-se a operação atualmente realizada [illegible] bits de "OPERAÇÃO". Se uma transferência é realizada utilizando-se caracteres co dificados em "ASCCII" (AMERICAN STANDARD FOR [illegible] INTERCHAN CE) por exemplo, o término da operação pode ser assinalado pela ocorrência de um caráter especial, chamado MARCA.

Esta situação é assinalada pelo bit "TRANSFERÊNCIA POR MARCA", sendo que a marca escolhida neste caso é o caráter "line-feed", de código $(0A)_{16}$.

Determinados eventos ocorridos na transferência , como por exemplo, a ocorrência de um erro [illegible] [illegible]cados no bit " CONTROLE".

TABELA DE ESTADOS DOS DISPOSITIVOS (TED)

A iniciaçao de uma operaçao de entrada ou saída é reali
zada por um primitivo especial (IES) que atualiza convenientemen-
te a TABELA DE ESTADOS DOS DISPOSITIVOS em funçao dos parâmetros
emitidos. Além deste, primitivos para alteraçao (AES) e leitura
(LES) do estado do dispositivo são implementados.

A continuaçao da operaçao realizada pela rotina CES (fig.
2.11) consiste de uma sequência de testes dos bits de estado da
linha correspondente à interface que provocou a interrupçao, trans
ferência de novos dados em funçao do CONTADOR ou da MARCA, indica
ção de um fim de operaçao, e finalmente a restauração do "STATUS"
da UCP e transição desta para o estado NORMAL a cada interrupçao.

Uma vez que a CES é comum a várias interfaces, é neces-
sário que as instruções de entrada e saída sejam montadas a cada
interrupçao, de acordo com o número da interface sendo tratada.
Além disso, a necessidade de um grande número de testes, o salva-
mento e recuperação do "STATUS" da U.C.P. podem tornar o atendimen
to dessas interrupções demorado o suficiente para prejudicar o de
sempenho do sistema quando vários dispositivos forem operados si-
multâneamente. Entretanto, a economia de memória proporcionada por
este esquema podem tornar essa deficiência suportável em uma ver-
sao mais simples do sistema operacional. Uma melhoria pode ser ob
tida substituindo-se a CES por rotinas especializadas para cada
interface que exigir uma velocidade mais alta no tratamento de suas
interrupções.

Pode-se observar que a rotina CES não provoca a suspen-
são do processo em execução, e ainda, que o final da operaçao de
um dispositivo (transferência de até 256 caracteres) é apenas as-
sinalada na tabela de estados correspondente à interface, devendo
o processo interessado nesta transferência tomar a iniciativa de
consultar este evento por meio de primitivos especiais (LES).

2.6.5.1 - PRIMITIVOS DO CANAL CONCENTRADOR DE ENTRADA E SAÍDA

A iniciaçao e acompanhamento das operações de entrada e
saída para os dispositivos do canal concentrador é possibilitada
pelos primitivos INICIA ENTRADA E SAÍDA (IES), LE ESTADO DO DISPO
SITIVO (LES) e ALTERA ESTADO DO DISPOSITIVO (AES).

CES

CONFIGURA INSTRUÇÕES DE E/S

FIM (F)

1 CET

0

CONT

0 CEF

≠0

ENTRADA SAIDA

SAIDA

TRANSFERE NOVO DADO PARA INF.

TIPO

MARCA

FIM-LINHA

SIM

CONT=0

CET

BINARIA

NÃO

ENTRADA

TRANSFERE DADO DA INTERFACE

TIPO

MARCA

ELIMINA BIT DE PARIDADE

BINARIA

CTN

FIM-LINHA

SIM

CEF

NÃO

CEF

CTC

CTA

ATUALIZA CONTADOR E ENDEREÇO

CET

CEF
BIT F=1
PIT=0
CET
PDI=0
RESTAURA
"STATUS"
PUL
CTC
LIGA FF DE
CONTROLE
INTERFACE
R.S
CTN
IGNORA
OS
SIM
NÃO
CEI
CEM
ZERO
SIM
NÃO
CTC
CET
DECREMENTA
CONTADOR
CONT
BIT C=1
CONT ≠0
SIM
R.S
NÃO
CEM
CEF

FIG. - 2.11

Os códigos de identificação são $(03)_{16}$, $(05)_{16}$ e $(04)_{16}$ respectivamente, enquanto que os demais parâmetros têm significado especial para cada um deles.

a) LÊ ESTADO DO DISPOSITIVO (LES)

O primitivo LES (j), onde j indica o número da interface em questão é codificado da seguinte forma :

CSP
5
j
E

Após a execução do primitivo, o valor do parâmetro E é atualizado com o conteudo dos bits EST (j) (fig. 2.10) e devendo ser interpretados da seguinte forma :

C = 1 : Dispositivo fora de operação ou transferência realizada com erro.

F = 1 : Última operação realizada e dispositivo desocupado.

OP : Operação realizada

b) ALTERA ESTADO DO DISPOSITIVO (AES)

Este primitivo AES (j, s, e, c) tem por finalidade preencher a linha da tabela do descritor do dispositivo j com os valores do estado (EST (j)), endereço (E_1 (j), E_0 (j)) e contador (CONT (j)) dados pelas constantes s, e, c respectivamente.

CSP

4

i

E

E_I

E_0

C

Além do preenchimento da tabela,nenhuma outra providência acompanha a execução desse primitivo.

c) INICIA ENTRADA E SAÍDA (IES)

IES (j) inicia condicionalmente a operação de entrada e saída no dispositivo j de acordo com as informações disponíveis na TABELA DE DESCRITORES DOS DISPOSITIVOS correspondentes a esse dispositivo j.

CSP

3

j

A condição necessária para que as instruções de entrada e saída sejam enviadas ao dispositivo é que o bit F de fim de operação seja igual a 0 (zero), de outra forma, é feito igual a 1 (um) e a execução do primitivo é terminada. O bit C determina, se igual a 1, que em uma entrada de dados, os caracteres nulos iniciais sejam ignorados.

Além desses primitivos, a alteração da prioridade no atendimento das interrupções pode ser conduzido por meio de um primitivo especial "TROCA DE PRIORIDADES" TRP (i, j) (2.2).

Este primitivo é codificado como :

CSP

$(C)_{16}$

i

j

A execução deste causa uma troca de posições dos tes-
tes de pedidos de interrupção das interfaces de números i e j rea
lizados na "ROTINA DE DESCOBRIMENTO DE INTERRUPÇÕES" (RDI).

O uso deste primitivo permite que se analise a influên
cia da prioridade de uma interface na velocidade de operação dos
dispositivos de entrada e saída quando utilizados simultâneamen-
te.

2.6.6 - COMUNICAÇÃO ENTRE PROCESSOS

A comunicação entre os processos se realiza mediante
uma troca de MENSAGENS.

Uma mensagem será definida como sendo um conjunto de
palavras consecutivas de número variável indicadas por endereço
disponível no NÚCLEO. Estas MENSAGENS ocupam posições da memória
reservadas por um processo, sendo que o controle e interpretação
de cada MENSAGEM esta a cargo do respectivo processo.

Ao NÚCLEO está atribuida a função de transmitir essas
MENSAGENS por meio de manipulação dos endereços das MENSAGENS.

Para esse efeito o NÚCLEO mantém uma lista de endere-
ços de MENSAGENS, estando em cada um dos quais assinaladas as
idenficações dos processos REMETENTE (R), e DESTINATÁRIO (D), con
forme a fig. 2.12 .

A TABELA DE DESCRITORES DE MENSAGENS é constituida por
15 linhas, possibilitando portanto a manipulação de até 15 men-
sagens simultâneas, linhas estas encadeadas por meio de uma lis-
ta ligada, cujo primeiro elemento é indicado pela variável PMU,
e ainda por uma lista de posições vazias encadeadas a partir de
PML. Os apontadores destas listas ocupam os quatro bits mais sig
nificativos da coluna TML.

Na coluna TMD comparecem as identificações dos proces-
sos envolvidos com a mensagem indicada pelo endereço END 1, END 0
nas colunas TML (4 bits menos significativos) e TMJ respectivamen
te.

TABELA DE DESCRITORES DE MENSAGENS

	TMD		TML		TMJ
0			PMU	PML	
1	D	R	PLM	END1	END 0
F					

Fig. - 2.12

2.6.6.1 - PRIMITIVOS PARA COMUNICAÇÃO ENTRE PROCESSOS

Três primitivos foram incluidos para a troca de MENSAGENS entre os processos, além de fornecer um método de sincronização entre eles.

a) ENVIA MENSAGEM (EMS)

O primitivo EMS (N, E_1, E_0), codificado como :

CSP

$(9)_{16}$

N

E_1

E_0

onde E_1, E_0 designa o endereço (na área de memória do processo remetente) de uma MENSAGEM a ser enviada ao processo N (destinatário).

A execução do primitivo se processa, em primeiro lugar ativando o processo destinatário (o processo N é colocado no estado ATIVO) se este estiver no estado ESPERA. Em segundo lugar, os valores de E_1, E_0 bem como D obtido através de N, e R obtido através de PEX (identificador do processo em execuçao) sao lançados na primeira linha disponível da TABELA DE DESCRITORES DE MENSAGENS.

Após a execução do EMS, o controle da U.C.P. retorna ao processo remetente.

Caso o processo identificado por N não exista, ou não haja linha disponivel nessa tabela, o processo remetente sofre um "ATRASO", isto é, o seu "STATUS" é atualizado de maneira quando for acionado novamente repita este mesmo primitivo.

Isto é realizado subtraindo-se duas unidades no CONTADOR DE INSTRUÇOES do processo remetente por ocasião da execuçao do primitivo e acionando-se o próximo processo em estado ATIVO.

b) AGUARDA MENSAGEM DE UM PROCESSO

AMS (N_I, E_I, E_0) codificado como :

CSP

$(A)_{16}$

N

E_I

E_0

transfere para as posições indicadas por E_I, E_0 o endereço de uma MENSAGEM enviada pelo processo N ao processo que promove a execução do primitivo. Este endereço é retirado da primeira linha da TABELA DE DESCRITORES DE MENSAGENS na qual os valores D e R são iguais as identificações do processo destinatário (PEX) e remetente (N), sendo em seguida lançada na lista vazia da tabela.

Caso o par D, R acima descrito não seja encontrado na TDM ou o processo remetente não existir (N), o processo que promoveu a execução de AMS é colocado no estado de ESPERA ao mesmo tempo que tem seu "STATUS" atualizado de maneira que quando acionado novamente promova a execução deste mesmo primitivo. Somente neste última situação, um próximo proces[illegible]ionado.

c) PRÓXIMA MENSAGEM (PXM)

O primitivo PXM (E_I, E_0) é assim codificado :

CSP

$(D)_{16}$

E_I

E_0

Este primitivo é processado de modo análogo ao AMG. Entretanto o controle é retornado ao processo que emitiu o primitivo, existindo ou não MENSAGEM para

No caso de não existir, uma MENSAGEM inválida é transmitida (bit 7 de E_1 = 1) ao processo.

d) AGUARDA MENSAGEM (AMG)

O primitivo AMG (E_1, E_0) codificado como :

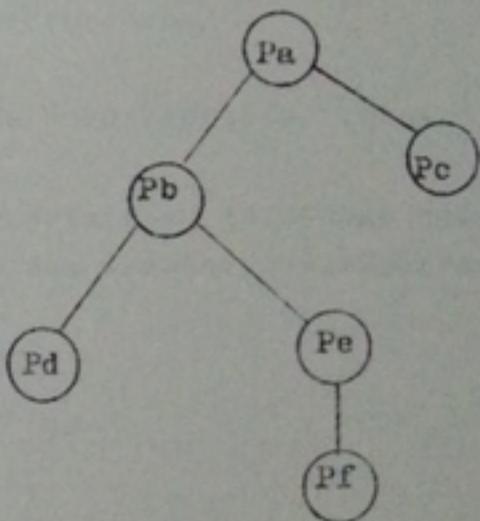
CSP

$(B)_{16}$

E_1

E_0

é processado de modo análogo ao primitivo AMS, exceto que a MENSAGEM recebida pode ter sido enviada por qualquer processo em andamento no sistema. Para isso, na execução de AMG, somente o processo destinatário (D) é verificado na tabela TMD.

2.6.6.2 - SINCRONIZAÇAO DE PROCESSOS

Os primitivos EMS, AMS e AMG podem ser utilizados na sincronizaçao de dois ou mais processos pelo fato de que as transições de estados provocadas por estes primitivos proporcionam um meio de se controlar os instantes em que tais processos devam ser executados ou devam permanecerem em ESPERA, em função de eventos constituidos pelas MENSAGENS trocadas entre eles.

Assim, EMS assinala tal evento dando condições para que o processo destinatário entre em EXECUÇÃO, enquanto que AMS e AMG podem provocar o atraso necessário na espera deste evento.

Quando tais eventos nao envolvam diretamente uma MENSAGEM, o primitivo ALTERA ESTADO DO PROCESSO (2.6.7) pode ser utilizado com o propósito de sincronizaçao.

2.6.7 - CONTROLE DE UM PROCESSO

Uma vez que não se estabeleceu definitivamente a natureza da aplicação de um sistema a multiprocessos utilizando o NÚCLEO aqui definido, alguns primitivos que proporcionem a possibilidade de um processo exercer controle sobre outros processos foram implementados, assumindo-se algumas hipóteses, que em princípio não afetam a generalidade do NÚCLEO e podem em algum tipo de aplicação facilitar a construção de um sistema operacional.

A primeira hipótese é da existência de uma hierarquia entre os processos estabelecida na forma de uma "árvore", como ilustra a figura abaixo (3).

Assim sendo, um processo só pode exercer o controle sobre um processo diretamente subordinado, como por exemplo, o processo P_b só pode controlar diretamente os processos P_e e P_d (conforme a figura acima).

Essa hierarquia é implementada assinalando-se, para cada processo indicado na TABELA DE DESCRITORES DE PROCESSOS, uma lista de processos subordinados. Cada lista é indicada pelos 4 bits mais significativos (PRS) da coluna TDL na TABELA DE DESCRIÇÃO DOS PROCESSOS (2.6.2.1).

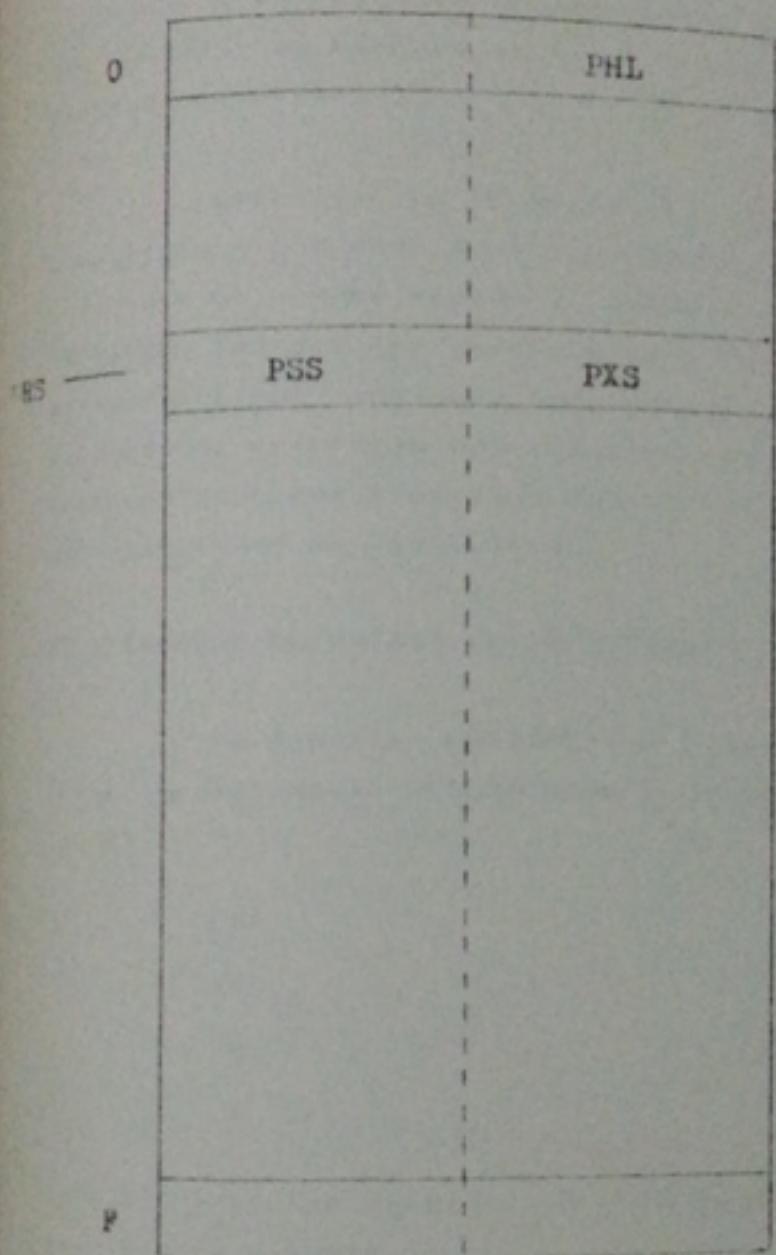
Estas listas sao implementadas de forma encadeada utilizando-se uma tabela TDH conforme figura 2.13. Cada elemento da lista contém a identificação do processo subordinado PSS e um indicador do próximo elemento desta lista PXS (PXS = 0 termina uma lista). O controle desta tabela é auxiliado por uma variável PHL indicando o início da lista de elementos livres da tabela.

A segunda hipótese admitida é a de que todos os recursos utilizados por um processo subordinado, exceto a U.C.P., são fornecidos e retirados pelo respectivo processo controlador.

2.6.7.1 - PRIMITIVOS DE CONTROLE

Foram escolhidos inicialmente quatro primitivos para o controle de processos.

a) ADICIONAR NOVO PROCESSO (CRP)

O primitivo ADICIONAR NOVO PROCESSO, CRP (N, S_0), é acompanhado dos seguintes parâmetros :

CSP
$(6)_{16}$
E_1
E_0

onde N é o código pelo qual o novo processo será referenciado, e $E_1 E_0$ é o endereço do DESCRITOR desse processo.

A execução desse primitivo é iniciada pela obtenção de uma posição na tabela TDP por meio das listas de processos e lista vazia implementados na coluna TDS. O índice da linha assim obtida passará a ser a identificação do novo processo e as colunas correspondentes de TDP são atualizadas com os valores de "STATUS" inicial (S_0), a coluna TDN com o código de referência (N) e a coluna TDH com a identificação do processo que promoveu a execução do primitivo CRP (PEX). Este último procedimento estabelece a relação de CONTROLADOR e SUBORDINADO que passará a existir entre estes processos.

0		PHL
PS	PSS	PXS
F		

Fig. - 2.13

O processo recém adicionado assume inicialmente o estado "PARADO", só podendo utilizar a U.C.P. quando o processo CONTROLADOR permitir.

Antes que um processo seja adicionado, os códigos que constituem o programa que irá implementar o novo processo, deverão ser colocados em alguma região de memória pertencente ao processo CONTROLADOR. Uma vez que não se dispõe "a priori" do endereço na memória principal onde cada programa deverá ser armazenado e não se dispondo de endereçamento relativo, um formato especial para a representação dos programas deverá ser utilizado. Estes aspectos são discutidos no capítulo 3.

b) ALTERAÇÃO DO ESTADO DE UM PROCESSO (AEP)

Um processo CONTROLADOR pode alterar o estado de um processo imediatamente SUBORDINADO mediante o uso do primitivo AEP (N, E).

CSP

$(7)_{16}$

N

E

onde N é o código do processo SUBORDINADO e E é o novo estado que este deverá assumir.

$E = (20)_{16}$: PARADO

$E = (40)_{16}$: ESPERA

$E = (80)_{16}$: ATIVO

O novo estado E é armazenado na linha correspondente ao código N da coluna TDS. Se o processo de código N não for SUBORDINADO direto do processo que emitiu o primitivo, este sofre um "ATRASO".

c) REMOÇÃO DE UM PROCESSO (RMP)

Um processo N, bem como todos os processos adicionados por este e seus subordinados, são removidos através da execução do primitivo RMP (N).

CSP
$(8)_{16}$
N

onde N é o código do primeiro processo a ser removido.

A remoçao de um processo consiste na liberação da identificação associada ao processo e a consequente atualização da TDP com o lançamento da linha contendo o descritor do processo na lista vazia.

Esse procedimento é realizado com o auxílio da coluna TDH na busca das identificações dos processos "descendentes" do processo removido.

Como nos outros casos, a inexistência do processo N ou a nao autorizaçao para a execução do primitivo RMP, causam um "ATRASO" ao processo emitente.

d) LEITURA DO DESCRITOR

A leitura do DESCRITOR de qualquer processo pode ser realizada mediante o primitivo LDP (N, E_1, E_0).

CSP
$(E)_{16}$
N
E_1
E_0

Por meio deste primitivo, o endereço do DESCRITOR do processo N é transferido para E_1, E_0, permitindo assim que incluíve o "STATUS" do processo possa ser manipulado.

CAPÍTULO 3 - UM SISTEMA BÁSICO DE CONTROLE PARA O "PATINHO FEIO"

3. UM SISTEMA BÁSICO DE CONTROLE PARA O "PATINHO FEIO"

De acordo com a introdução deste trabalho, o Sistema Básico de Controle desenvolvido para o "PATINHO FEIO" tem por finalidade ilustrar uma aplicação do NÚCLEO BÁSICO descrito no capítulo 2.

Essa aplicação consiste na execução de vários processos em paralelo.

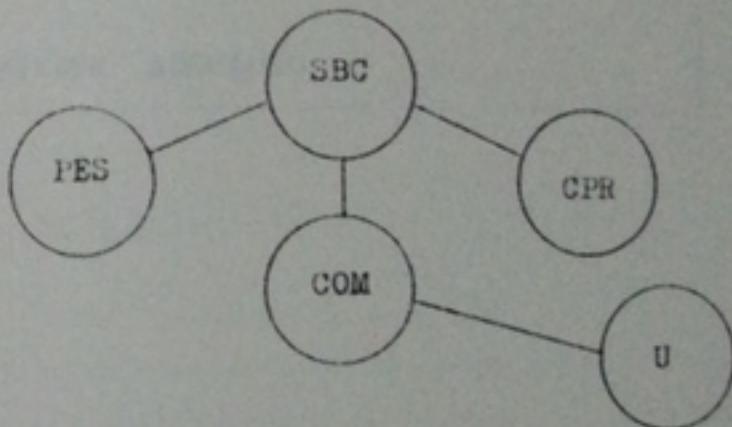
Alguns destes processos terão aplicações específicas, como exemplo, o controle das tarefas de ENTRADA/SAÍDA ou carregamentos de programas, enquanto que os outros poderão ser úteis na simulação de outros processos que poderão vir a ser criados conforme os recursos do computador o permitirem. Essa simulação permite criar-se situações típicas dos processos simulados, principalmente com respeito à sincronização e comunicação. A observação do comportamento de um processo simulador pode estabelecer alguns critérios no projeto do processo propriamente dito.

3.1 - ORGANIZAÇÃO DO SISTEMA BÁSICO DE CONTROLE

Este Sistema Básico de Controle utiliza o "PATINHO FEIO" com a configuração:

a) Dispositivos de ENTRADA/SAÍDA

LEITORA DE FITA DE PAPEL PERFURADO
PERFURADORA DE FITA
TELEIMPRESSORA
IMPRESSORA DE LINHAS
LEITORA DE CARTÕES PERFURADOS

Estes dispositivos são interligados no "CANAL CONCENTRADOR" do NÚCLEO.

b) Memória

4096 palavras de 8 bits

c) Software

NÚCLEO BÁSICO

É constituído por quatro processos básicos, o Sistema Básico de Controle (SBC) de código 1 (um), um processo Controlador de Entrada/Saída (PES) com código 2, um processo Controla Carga de Programas (CPR) com código 3 e um processo de Comunicação com o operador (COM) de código 4.

Estes processos estão organizados em uma hierarquia, esquematizada abaixo, que possibilita ao SBC o exercício do controle sobre os demais processos. Qualquer outro processo U no sistema será controlado diretamente pelo processo COM.

3.1.1 - ORGANIZAÇÃO DA MEMÓRIA PRINCIPAL

A Memória Principal foi organizada inicialmente como mostrado na figura 3.1. Apresenta-se subdividida em 16 "PÁGINAS" de 256 palavras cada uma apenas com o objetivo de ressaltar a proporção de cada programa em relação ao total. Assim, o NÚCLEO ocupa as primeiras oito "páginas", o SBC, PES e CPR as quatro "páginas" seguintes e COM as quatro "páginas" restantes. Procurou-se manter essas dimensões como limites máximos a fim de proporcionar ao processo COM uma área relativamente grande uma vez que esta deverá conter os demais processos administrados pelo COM. Isto é possível uma vez que PES e CPR não admitem outros processos subordinados.

A "página" 15 contém a partir do endereço $(F00)_{16}$ ao programa CARREGADOR ABSOLUTO que foi considerado como integrante

NUCLEO BASICO
SBC CPR PES
COM
CARREGADOR ABSOLUTO

Fig. - 3.1

FORMATO DOS DADOS (ABSOLUTO)

N	E1	E0	D1	D2	...	Di	...	Dn	T

Fig. - 3.2

do NÚCLEO.

As funções deste CARREGADOR ABSOLUTO serão mostradas em 3.1.2.

3.1.2 - INICIAÇÃO DO SISTEMA BÁSICO DE CONTROLE

A iniciação do Sistema Básico de Controle consiste em introduzir na Memória Principal, os códigos constituídos pelo NÚCLEO, PES, COM e CPR dispostos como mostrado em 3.1.1.

Estes códigos fazem parte de um arquivo em fita de papel perfurado no formato ABSOLUTO. Deste modo pode-se utilizar o CARREGADOR ABSOLUTO nesta operação, após o que a execução do Sistema Básico de Controle irá iniciar-se no processo SBC.

3.1.2.1 - O PROGRAMA CARREGADOR ABSOLUTO

O CARREGADOR ABSOLUTO ocupa 128 últimas posições da Memória Principal, estando estas protegidas sempre que uma chave do painel estiver ligada. Assim sendo, nenhuma operação de escrita será realizada em posições a partir do endereço $(F80)_{16}$.

A função do CARREGADOR ABSOLUTO é transportar para a Memória Principal os códigos de um programa em formato ABSOLUTO contidos em uma fita de papel perfurada, utilizando a LEITORA DE FITAS PERFURADAS.

Este formato ABSOLUTO, esquematizado na figura 3.2, é constituído por palavras (8 bits) contendo:

N Número de palavras do arquivo

E_1,E0 Endereço de armazenamento na Memória Principal

P_0 a P_n Códigos a serem armazenados

T   Teste de Soma

O "Teste de Soma" (CHECKSUM) representa a soma dos conteúdos de todas as palavras produzidas no arquivo, sem levar em conta os transbordamentos.

Na leitura do arquivo, a mesma operaçao de soma é realizada e o resultado é comparado com T, proporcionando um método de Verificaçao da existência de erro de leitura.

No caso de erro de leitura, o CARREGADOR ABSOLUTO indica através do MOSTRADOR DO ACUMULADOR do painel um código diferente de zero.

Se o transporte do programa for realizado com sucesso (MOSTRADOR DO ACUMULADOR = 0), os valores E_1, E_0 sao colocados nas posiçoes 6 e 7 da Memória e é executada uma instruçao de PARE.

Acionando-se o botao PARTIDA, o CARREGADOR ABSOLUTO executa um desvio incondicional para a posiçao 6 iniciando portanto a execuçao do programa que foi armazenado.

As 128 posiçoes constituídas pelo CARREGADOR ABSOLUTO podem ser armazenadas na Memória Principal por meio de pequenos programas chamados "PRÉ-CARREGADORES" armazenados através das chaves DO PAINEL. A proteçao dessas 128 posiçoes de Memória permite manter-se o CARREGADOR ABSOLUTO por longos períodos de tempo inalterado. Em se tratando de Memória de Núcleo de Ferrite, este poderá ser utilizado mesmo que o computador tenha sido desligado. Entretanto acidentes de operaçao do computador e mesmo ruídos produzidos por ocasiao de um desligamento do computador, por exemplo, fazem com que a utilizaçao de uma mesma cópia na Memória por longos períodos nao confiável.

Uma soluçao seria a utilizaçao de uma interface especial capaz de realizar essa operaçao de transporte de programas para a Memória Principal de uma maneira automática por "Hardware".

3.2 - DESCRIÇÃO DO PROCESSO SBC

O processo SBC é executado somente após o armazenamento do Sistema Básico de Controle na Memória Principal por meio do CARREGADOR ABSOLUTO.

Tem por finalidade iniciar as operações das interfaces de interrupção síncrona (interface 1) e de RELÓGIO DE TEMPO REAL (interface 2), criar os processos PES, COM e CPR e ativar o processo COM.

A estrutura dos dados do NÚCLEO é armazenada com os valores iniciais convenientes, de tal maneira que por ocasião da execução de SBC, o NÚCLEO admite a existência apenas do processo SBC.

Após a ativação do processo COM, o processo SBC é colocado no estado PARADO, não sendo mais reativado. A figura 3.1 apresenta um diagrama do processo SBC.

3.2.1 - INICIAÇÃO DA INTERFACE 1

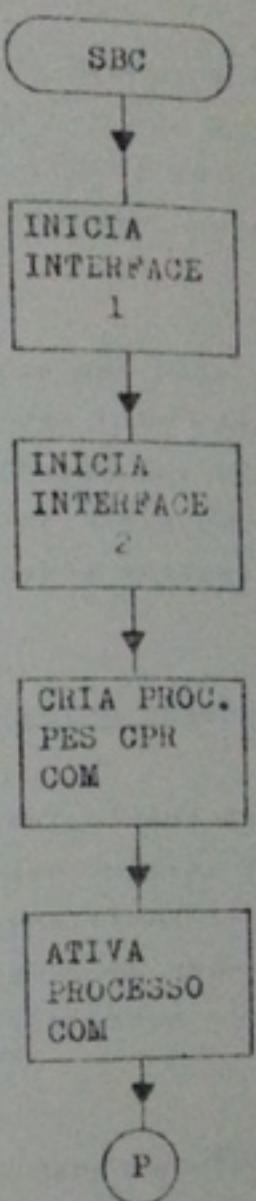
A iniciação da interface 1 consiste em fazer o sinal ESTADO da interface tomar o nível lógico "1" $(EST)_i \leftarrow 1)$ conforme mencionado em 2.3, e ainda preparar o sinal de PEDIDO DE INTERRUPÇÃO para a primeira interrupção síncrona que irá ocorrer $(PDI)_i \leftarrow 0)$.

3.2.2 - INICIAÇÃO DO RELÓGIO DE TEMPO REAL

O RELÓGIO DE TEMPO REAL é iniciado com um período de 100 ms, correspondente a 2 unidades de tempo (2.6.4). Entretanto o seu pedido de interrupção é apenas liberado após a execução do primitivo AEP que coloca o processo SBC no estado PARADO.

SBC
INICIA INTERFACE 1
INICIA INTERFACE 2
CRIA PROC. PES CPR COM
ATIVA PROCESSO COM
P

Fig. - 3.3

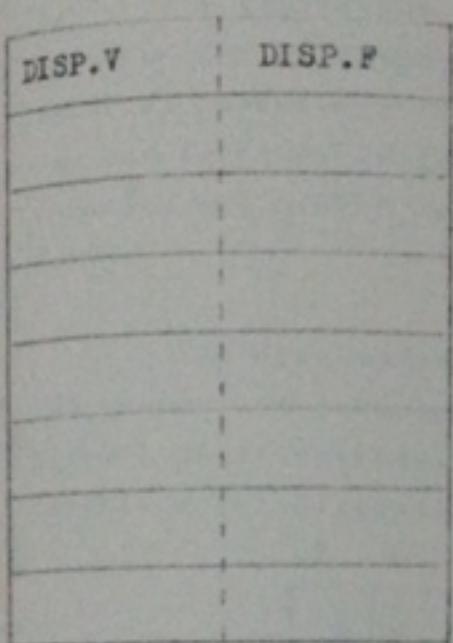
3.2.3 - OBSERVAÇÕES SOBRE O PROCESSO SBC

A existência do processo SBC apresenta neste caso um caracter mais formal, uma vez que as operações que realiza poderiam ser conduzidas pelo NÚCLEO em resposta a um pedido de interrupção do PAINEL, por exemplo, ou pelo próprio CARREGADOR ABSOLUTO.

Uma modificação que pode ser útil em alguma aplicação, consiste em fazer com que este inicie a estrutura de dados do NÚCLEO, sempre que reacionado por uma interrupção do PAINEL (botão de INTERRUPÇÃO). Esta reiniciação da estrutura de dados pode acompanhar o teste de novas rotinas incorporadas no NÚCLEO.

3.3 - DESCRIÇÃO DO PROCESSO CONTROLADOR DE ENTRADA/SAÍDA

O processo controlador das operações de Entrada / Saída (PES) tem por finalidade iniciar e supervisionar as operações de Entrada/Saída dos dispositivos ligados ao CANAL CONCENTRADOR (2.6.5), assinalando os eventos associados a esses dispositivos aos processos que tenham requerido a utilização dos mesmos.

3.3.1 - ESTRUTURA DOS DADOS DO PROCESSO PES

Os dispositivos do CANAL CONCENTRADOR são referenciados por números (0 a 7) que constituem as chamadas UNIDADES LÓGICAS (UL).

A utilização de UNIDADE LÓGICA permite por exemplo que um mesmo dispositivo seja referenciado por UNIDADES LÓGICAS diferentes, ou ainda, oferece a possibilidade de, sem alterar o programa, um processo poder utilizar um outro dispositivo usando a mesma UL, como será visto em 3.3.2.

Com o propósito de tornar flexível a associação entre UNIDADE LÓGICA, dispositivo e interface, o processo PES mantém três tabelas de 8 palavras cada, designadas por TUL, TDD e TPN.

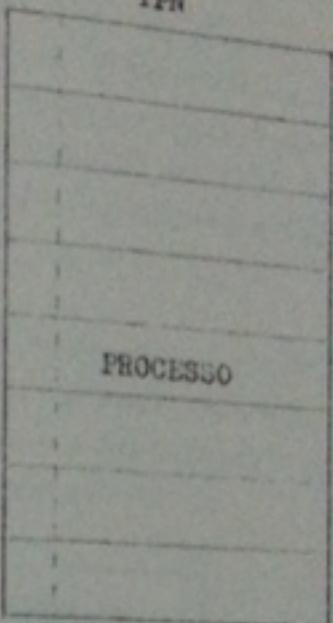
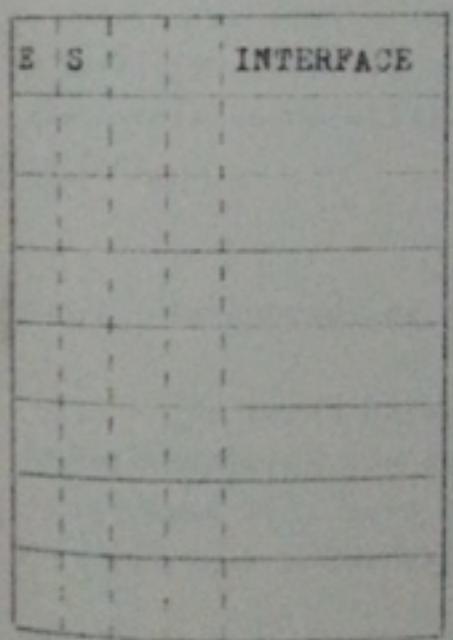
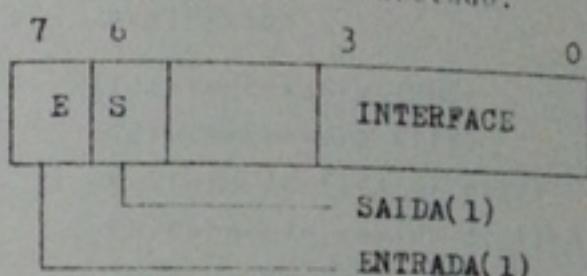



FIG. - 3.4

A tabela TUL associa a uma dada UL (índice na tabela) dois números de dispositivos, um fixo (DISP.F) e um variável (DISP.V) conforme a figura 3.4. O número DISP.V é utilizado para indicar (através de indexação) um elemento da tabela TDD ou TPN, enquanto que DISP.F serve apenas para restaurar os valores DISP. V.

Em correspondência a cada dispositivo (DISP.V) a tabela TDD contém uma pequena descrição das características deste dispositivo, se de entrada, saída ou ambos. Especifica ainda a qual interface tal dispositivo está conectado.

Por fim, a tabela TPN contém em correspondência a um dispositivo (DISP.V) o código do processo que o utiliza. Caso não esteja sendo utilizado, este elemento de TPN conterá o código $(FF)_{16}$.

3.3.2 - PARÂMETROS DE COMUNICAÇÃO DO PROCESSO PES

Os tipos de operação a serem efetuados nos dispositivos de Entrada/Saída são designados mediante o envio de uma MENSAGEM ao processo PES. Esta MENSAGEM tem os seguintes formatos, conforme a operação a ser exercida por PES:

a) MUDANÇA DE UNIDADE LÓGICA

MENSAGEM R F U_x U_y

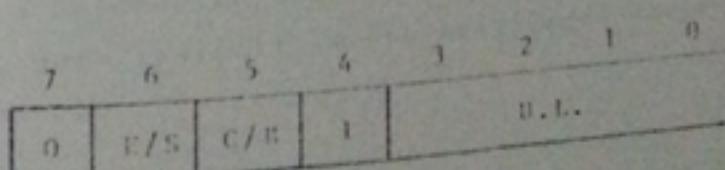
Onde as quatro palavras que compõem esta MENSAGEM são constituídas por:

R-Código do processo remetente

F-Função, $F=(80)_{16}$

U_x-Unidade lógica
U_y-Unidade lógica

Este comando faz com que o dispositivo (DISP.F) associado à UNIDADE LÓGICA U_y passe a ser o dispositivo (DISP.V) associado à UNIDADE LÓGICA U_x.

Na versão inicial do Sistema Básico de Controle, a correspondência entre DISP.F e os dispositivos periféricos utilizados é a seguinte:

DISP.F	PERIFÉRICO
0	Teleimpressora (DEC)
1	Teleimpressora (TTY)
2	Leitora de fita de papel
3	Perfuradora rápida de fita de papel
4	Impressora de linhas
5	Leitora de cartões

b) OPERAÇÃO DE ENTRADA/SAÍDA

MENSAGEM R F E_1 E_0 C

Neste caso, R será ainda o Código do processo remetente, E_1 e E_0 conterão o endereço na memória a partir do qual os dados deverão ser transmitidos e C indicará o número desses dados (número de palavras de 8 bits).

A função F entretanto deverá ser codificada da seguinte forma:

7	6	5	4	3	2	1	0
0	E/S	C/B	I	U.L.			

E/S-Entrada (0) ou Saída (1)
C/B-Por caracteres (0) ou Binária (1)
I-Ignora zeros iniciais (0)
U.L-Unidade Lógica

A Entrada ou Saída de dados por caracteres é feita por marca (2.6.5) sendo os caracteres codificados em ASCII e a MARCA indicada pelo código $(0A)_{16}$ ("Line.Feed"). Neste caso, o contador C indicará apenas o número máximo de caracteres que podem ser transferidos.

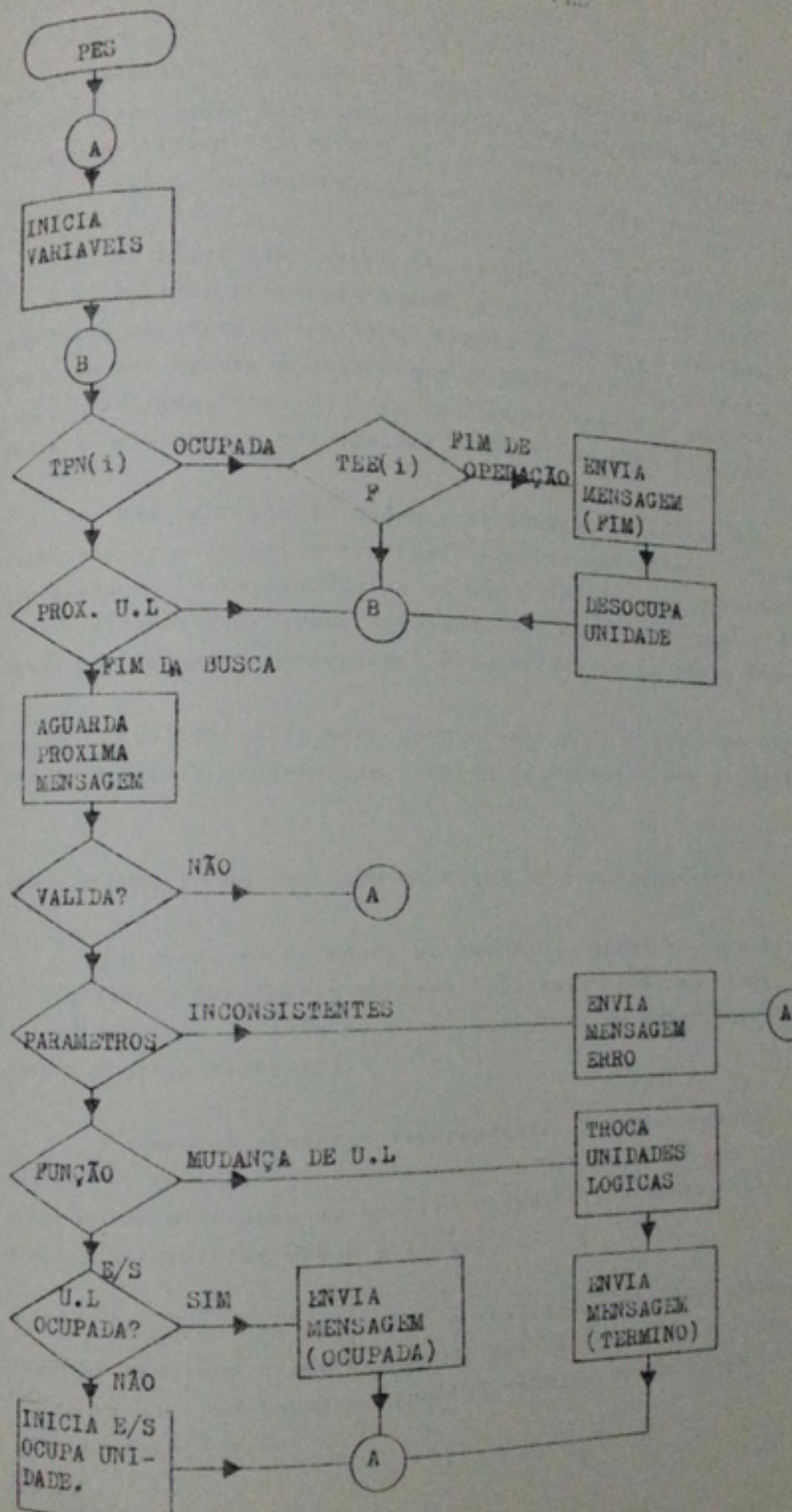
Na Entrada ou Saída Binária, a transferência é controlada apenas pelo contador C.

Em uma Entrada por Caracteres ou Entrada Binária com bit I=1, os caracteres nulos ("NULL") que precedem os dados são ignorados.

As respostas às MENSAGENS enviadas ao processo PES são constituídas por MENSAGENS de uma palavra com os seguintes significados:

RESPOSTA

7	6	5	4	3	2	1	0
E	U	0	0	0	0	0	0

E	U	
0	0	Operação realizada
0	1	Unidade Lógica sendo utilizada por outro processo
1	1	Erro por inconsistência da Mensagem

Após o envio de uma MENSAGEM ao processo PES, o processo remetente deve aguardar uma MENSAGEM de PES codificada como indicada acima.

Caso a Unidade Lógica referenciada estiver sendo ocupada, o processo pode insistir em sua utilização enviando novamente a mesma mensagem.

3.3.3 - ESTRUTURA DO PROCESSO PES

O processo PES permanece sempre no estado ATIVO, realizando, quando em execução, basicamente um teste de todas posições da tabela TUL a fim de verificar o término de alguma operação

DIAGRAMA DO PROCESSO PES

Fig. - 3.5

ção de ENTRADA/SAÍDA realizada nos dispositivos constantes desta tabela (fig. 3.5). Caso uma operação termine, fato este constatado pelo primitivo LES (2.6.5.1), uma MENSAGEM de "Operação realizada" é enviada ao processo que utilizou o dispositivo.

No final dos testes dos dispositivos, a próxima MENSAGEM é solicitada e testada quanto a sua validez. Se houver nova MENSAGEM, um teste é realizado quanto à sua consistência, como por exemplo, quanto à existência da Unidade Lógica, se a operação é permitida, etc. No caso da inconsistência o processo remetente é avisado por meio de uma MENSAGEM de ERRO (3.3.2).

Uma operação de Mudança de Unidade Lógica só não será realizada se o processo remetente não for autorizado, enquanto que uma operação de Entrada/Saída só não é iniciada se a Unidade Lógica pedida estiver sendo utilizada por outro processo. Nestes casos uma MENSAGEM conveniente é retornada ao processo remetente.

Realizadas as operações requeridas, o processo PES inicia novamente a pesquisa do término de operação nos dispositivos.

3.4 - DESCRIÇÃO DO PROCESSO DE CARGA DE PROGRAMAS (CPR)

O processo de CARGA de PROGRAMAS (CPR) tem por finalidade de realizar o transporte de programas cujos códigos estão contidos em arquivos de fita de papel perfurado ou cartões perfurados para a Memória Principal.

Sendo um processo independente, podendo portanto ser ativado por qualquer outro processo do S.B.C., este não realiza nenhuma administração da Memória Principal, estando esta função a cargo do processo que a utiliza.

A carga dos programas é realizada incondicionalmente, supondo os arquivos estruturados em dois formatos, o formato ABSOLUTO (3.2.2) e um formato especial chamado RELATIVO que será descrito no item 3.4.1.

3.4.1 - PARÂMETROS DE COMUNICAÇÃO DO PROCESSO CPR

A ordem ao processo CPR para que este realize a carga de um programa é realizada mediante o envio de uma MENSAGEM, cuja interpretação dos parâmetros dependerá do formato especificado.

a) FORMATO ABSOLUTO

MENSAGEM R F U.L. E_1 E_0

onde R - Código do processo remetente

F - Formato, igual a 0 (zero)

U.L. - Unidade Lógica de origem do arquivo

E_1, E_0 - Endereço na Memória Principal para a carga do programa

O endereço E_1, E_0 é conferido com aquele existente no arquivo ABSOLUTO (3.1.2.1) disponível na Unidade Lógica referida e caso for diferente, o pedido é rejeitado.

b) FORMATO RELATIVO

MENSAGEM R F U.L. B_1 B_0

R - Código do processo Remetente

F - Formato igual a 1 (um)

U.L. - Unidade Lógica de origem do arquivo

B_1, B_0 - Base de Relocação

Um arquivo de programa no formato RELATIVO é constituído por dois tipos de blocos, cada um com no máximo 128 palavras.

O bloco de IDENTIFICAÇÃO deve ser o primeiro constante no arquivo, seguido de um número indefinido de blocos de DADOS (fig. 3.6).

Um bloco de IDENTIFICAÇÃO, contém:

N - Número de palavras do bloco (complemento de dois)

I_2, I_1, I_0 - três caracteres ASCII identificando o programa

C - Código do programa

DD_1, DD_0 - Dimensão da área de dados do programa (número de palavras)

T - Teste de soma

Um BLOCO DE DADOS conterá:

N - Número de palavras do bloco

BE_1, I_0 - Endereço relativo de carga do bloco. (R=0 BASE DE DADOS, R=8 BASE DE PROGRAMA

$R_j, D_{0j}, D_{1j}, D_{2j}$ - elemento do bloco

T - Teste de Soma

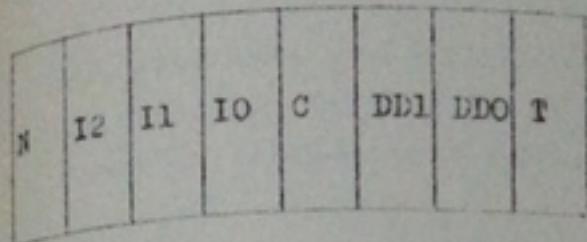
Cada elemento do BLOCO DE DADOS (R_1, D_0, D_1, D_2) é constituído por um CÓDIGO de RELOCAÇÃO R e 4 dados de uma ou duas palavras cada um, de acordo com o conteúdo do CÓDIGO DE RELOCAÇÃO. Este especifica a relocação de cada um dos dados por meio de dois bits.

CÓDIGO DE RELOCAÇÃO R_j R_{j0} R_{j1} R_{j2} R_{j3}

$R_j = (00)_2$ - Dados Absoluto (uma palavra)

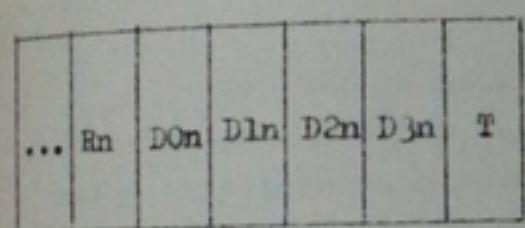
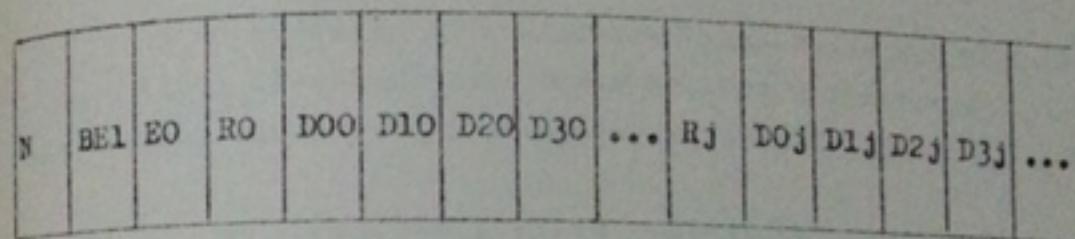
$R_{ji} = (01)_2$ - Dado relativo à área de Dados (2 palavras)

$R_{ji} = (10)_2$ - Dado relativo à Área de programa (2 palavras)

$R_{ji} = (11)_2$ - Final lógico do Bloco de Dados

$i = 0, 1, 2, 3$

Em resposta a um pedido de carga de programa, o processo CPE retorna ao processo remetente com uma MENSAGEM especificando, através de C uma das seguintes situações:

$C = 0$ - Operação não realizada por motivo de erro

$C \neq 0$ - Operação realizada, com C igual ao código do programa carregado de um arquivo RELATIVO ou C=0 se de um arquivo ABSOLUTO.

RESPOSTA C

N
I2
I1
IO
C
DD1
DDO
T
BLOCO DE IDENTIFICAÇÃO
BLOCO DE DADOS
N
BE1
EO
RO
DOO
D1O
D2O
D3O
...
Rj
DOj
D1j
D2j
D3j
...
...
Rn
DOn
D1n
D2n
D3n
T
n=NE
ESTRUTURA DOS DADOS DO PROCESSO CPR
BSE
PBL
ENE

Fig. - 3.6

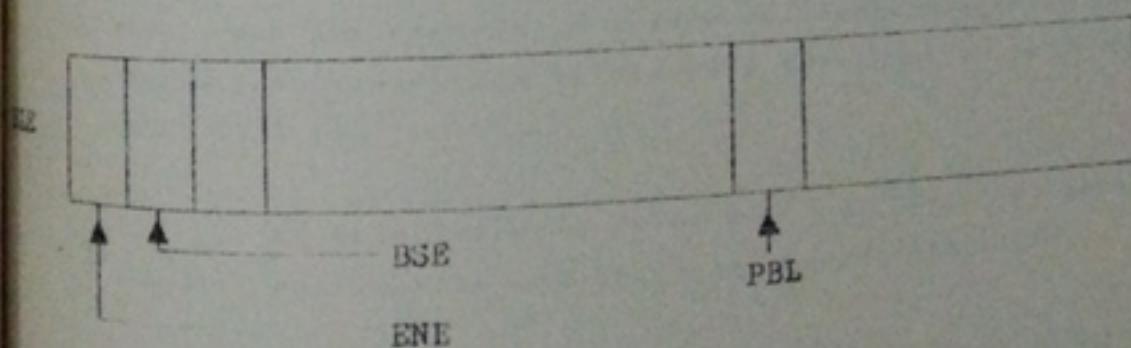
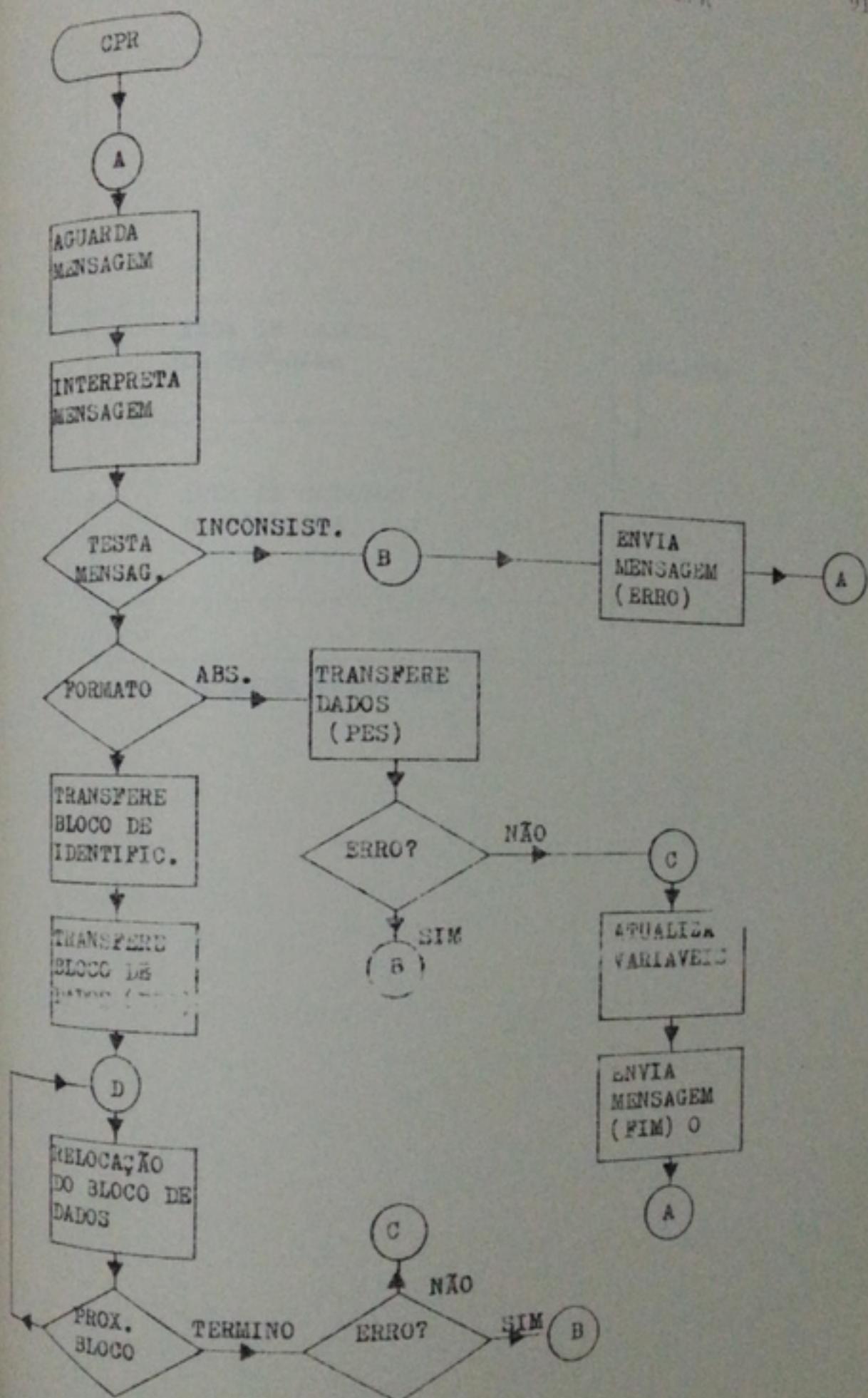



Fig. - 3.7

3.4.2 - ESTRUTURA DOS DADOS DO PROCESSO CPR

O processo CPR utiliza basicamente uma área de 128 palavras designada por BLE, cujos primeiros elementos são rotulados por ENE e BSE (fig. 3.7). A busca de elementos nesta área é realizada utilizando-se o indicador PBL.

3.4.3 - ESTRUTURA DO PROCESSO CPR E RELOCAÇÃO

O processo CPR inicia-se aguardando uma MENSAGEM qualquer (AMG). Recebida uma MENSAGEM, CPR a interpreta, realiza a operação requisitada e aguarda uma nova MENSAGEM (fig. 3.8).

As operações a serem realizadas consistem na transferência de um arquivo da Unidade Lógica especificada utilizando o processo PES.

No caso de arquivo no formato ABSOLUTO, é suficiente o envio de uma MENSAGEM ao processo PES com os parâmetros convenientes extraídos da MENSAGEM recebida por CPR, após o que este aguarda uma resposta de PES para verificar se a transferência foi bem sucedida.

Quando o formato for RELATIVO, a transferência dos dados é feita bloco por bloco na área BLE (fig. 3.7). Os parâmetros do bloco de identificação são armazenados em variáveis temporárias, como por exemplo, o código de identificação do programa que deverá ser transmitido ao processo remetente em resposta à MENSAGEM (3.4.1).

Na transferência de um bloco de Dados, no entanto, é necessário realizar-se uma operação chamada RELOCAÇÃO.

De acordo com o formato RELATIVO, um programa é constituído por uma ÁREA DE DADOS e uma ÁREA DE CÓDIGOS, (fig. 3.9). O endereço inicial da Área de Dados (BD) é especificado na MENSAGEM (B_1,B_0) enquanto que o início da Área de Códigos é calculada pela dimensão da Área de Dados especificada no Bloco de identificação do programa (DD_1, DD_0). Deste modo, $BP=BD+DD_1,DD_0$.

AREA DE DADOS
DO PROGRAMA

DD1,DDO

AREA DE CODIGOS
DO PROGRAMA

MEMORIA PRINCIPAL

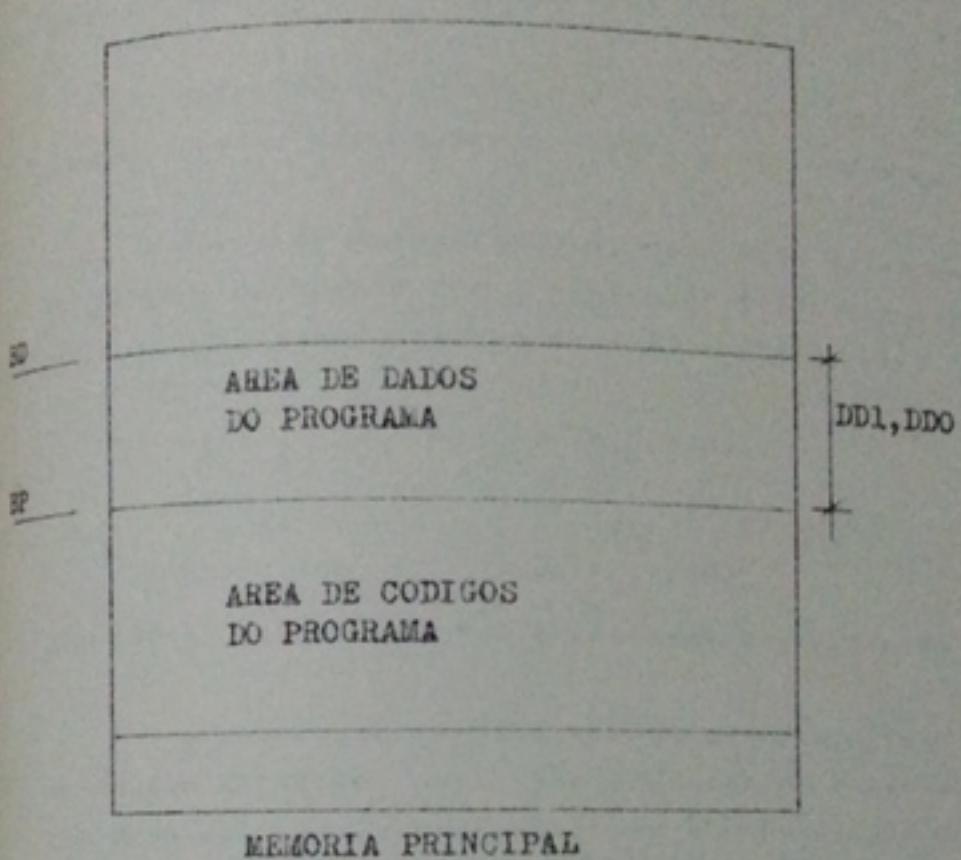



Fig. - 3.9

O armazenamento de cada bloco de Dados é feito a partir de um endereço calculado como: $E=BP+E_1,E_0+7$ ou $E=BD+E_1,E_0$ conforme a especificação B contida no início do bloco (3.4.1-b).

Calculado o endereço de armazenamento, os dados são armazenados consecutivamente a partir deste endereço E.

Antes do armazenamento, cada dado sofre uma modificação (RELOCAÇÃO) de acordo com o código de relocação correspondente. Assim, o dado a ser armazenado (DA) é obtido por:

$DA=D_{ij}$ se $R_{ji}=(00)_2$

$DA=D_{ij}+BD$ se $R_{ji}=(01)_2$

$DA=D_{ij}+BD$ se $R_{ji}=(10)_2$

para j=0,1...NE e i=0,1,2,3 conforme especificado no item 3.4.1-b

A Relocação permite que um mesmo programa constante de um arquivo RELATIVO, possa ser transferido e executado em qualquer área da Memória Principal sem a necessidade de se modificar o arquivo, fato este necessário quando se trata de um arquivo no formato ABSOLUTO, em virtude da ausência de ENDEREÇAMENTO RELATIVO no minicomputador "PATINHO FEIO".

Finalmente, o DESCRITOR do programa é atualizado nas 7 posições iniciais a partir da Base de programa (2.6.2.2).

3.4.4 - PRODUÇÃO DOS ARQUIVOS DE PROGRAMA

O desenvolvimento de um programa para o "PATINHO FEIO" é realizado em duas etapas. A primeira consiste na descrição em linguagem do MONTADOR RELOCAVEL (2) de um conjunto de unidades do tipo PROGRAMA PRINCIPAL, SEGMENTO e SUBROTINA que irão compor o programa.

O MONTADOR RELOCAVEL processa cada unidade independentemente, produzindo para cada uma um arquivo em formato RELOCAVEL, que será aqui chamado de MÓDULO RELOCAVEL. Cada módulo RELOCAVEL

é constituído por um bloco de identificação (BLOCO DE NOME), bloco de declaração de "PONTOS DE ENTRADA" (BLOCO ENT), bloco de declaração de referências externas ao módulo (BLOCO EXT), um conjunto de blocos descrevendo a unidade (BLOCO DE DADOS) e um bloco de finalização (BLOCO DE FIM).

MÓDULO
RELOCÁVEL   NOME   ENT   EXT   DADOS   ...   DADOS   FIM

Em um bloco NOME comparece o nome dado ao módulo, o tipo de unidade e a dimensão da ÁREA COMUM DE DADOS. O bloco ENT contém uma lista dos nomes das variáveis locais à unidade, declaradas na linguagem pelas pseudo-instruções ENT, enquanto que o bloco EXT conterá a lista dos nomes das variáveis que serão definidas em outras unidades. Tais nomes são declarados na linguagem fonte por meio das pseudo-instruções EXT (2).

Nos blocos de DADOS comparecem as instruções de máquina produzidas pelo MONTADOR RELOCAVEL. Cada instrução é acompanhada por um identificador especificando o tipo de instrução, e em particular tratando-se de uma instrução de referência, é especificado se o endereço da referência é relativo à ÁREA DE DADOS, à ÁREA LOCAL do programa ou a uma variável externa.

O módulo é terminado pelo bloco FIM especificando o endereço de execução do programa em se tratando de uma unidade do tipo PROGRAMA PRINCIPAL ou SEGMENTO. Os detalhes desta linguagem RELOCAVEL podem ser vistos na referência (9).

A segunda etapa consiste em reunir um conjunto de MÓDULOS RELOCAVEIS com o objetivo de produzir um único arquivo contendo o programa. Tal arquivo será chamado de MÓDULO DE CARGA.

A utilização de MÓDULOS RELOCAVEIS montados independentemente permite além da natural modularidade na programação (5), a possibilidade de se constituir uma BIBLIOTECA DE ROTINAS composta por uma coleção de MÓDULOS RELOCAVEIS que podem ser usados na produção de vários programas.

Além disso, uma vez que a linguagem RELOCAVEL é indepen-

dente da linguagem fonte utilizada, desde que um COMPILADOR de uma linguagem fonte qualquer produza os códigos objetos nesta mesma linguagem RELOCAVEL, é possível confeccionar-se programas obtidos a partir de rotinas escritas na linguagem do MONTADOR ou do COMPILADOR indistintamente.

A reunião de um conjunto de módulos RELOCAVEIS é basicamente a tarefa dos programas do tipo RELOCADOR-LIGADOR ou CARREGADOR-RELOCAVEL ("LINK-EDITOR", "RELOCATING-LOADER", ou "CORE-IMAGE BUILDER").

No "PATINHO FEIO" a producão de um MÓDULO DE CARGA a partir de um conjunto de MÓDULOS RELOCÁVEIS tem sido realizada pelo programa CARREGADOR-RELOCAVEL. As duas versões implementadas, descritas nas referências (9) e (17), produzem arquivos no formato ABSOLUTO (3.1.2.1).

Nesta produção, três funções básicas são realizadas pelo CARREGADOR-RELOCAVEL, a LOCALIZAÇÃO, RELOCAÇÃO e LIGACÃO dos MÓDULOS RELOCÁVEIS.

a) LOCALIZAÇÃO

O arquivo produzido, quando armazenado na Memória Principal (3.4), conterá os códigos e dados de cada módulo RELOCAVEL, localizadas sequencialmente a partir da ÁREA COMUM DE DADOS. A ÁREA COMUM DE DADOS, com início (BE) especificado pelo usuário tem uma dimensão estabelecida no módulo contendo a unidade PROGRAMA PRINCIPAL ou SEGMENTO. De acordo com a figura 3.10 conhecendo-se a dimensão da ÁREA COMUM DE DADOS, a ORIGEM OPP do PROGRAMA PRINCIPAL ou SEGMENTO (OPP) é estabelecida.

A determinação das ORIGENS de cada módulo RELOCÁVEL é realizada de forma análoga, conhecendo-se a dimensão do módulo anterior ($OS_1, OS_2, \ldots, OS_n$).

b) RELOCAÇÃO

Uma vez que por ocasião da produção dos MÓDULOS RELOCÁVEIS, a localização do módulo não é conhecida, as instruções de

BR
AREA
COMUM
DE DADOS
OPP
RE
PP ou SG
OS1
MRO
SUBR1
RC
OS2
MR1
SUBR2
RL
MR2
OSn
SUBRn
MRn

Fig. - 3.10

referência às variáveis locais do módulo contém endereços relati vos à origem deste, suposta igual a 0 (zero). A RELOCAÇÃO consis te portanto em corrigir esses endereços somando-se a estes a ori gem do módulo, estabelecida pelo método mencionado anteriormente.

Além das referências LOCAIS, uma instrução que referên cia a ÁREA COMUM DE DADOS sofre uma relocação utilizando-se a o rigem (EC) desta área. Na figura 3.10 estes tipos de referências estão indicados por RL e RC respectivamente.

c) LIGAÇÃO

A possibilidade de uma instrução referenciar variáveis contidas em outras unidades de programação constitui-se na manei ra pela qual estas unidades podem interagir. O conjunto de variá veis cujos nomes são declarados nos blocos de PONTOS DE ENTRADA (ENT) dos MÓDULOS RELOCÁVEIS constituem-se no conjunto das VARIÁ VEIS GLOBAIS.

As VARIÁVEIS GLOBAIS são definidas com endereços relati vos à origem dos MÓDULOS RELOCÁVEIS onde são declaradas no bloco ENT_j tornando-se portanto LOCAIS a esse MÓDULO em particular. Por este motivo, uma vez estabelecida a origem do MÓDULO, este parti cular subconjunto das VARIÁVEIS GLOBAIS terão seus endereços ab solutos definidos por meio de RELOCAÇÃO.

Definidos os endereços absolutos de todas as variáveis Globais, após a localização dos MÓDULOS (a), uma instrução que referencie o nome de uma variável externa poderá ter este nome substituído pelo endereço absoluto calculado anteriormente.

Esta operação (LIGAÇÃO) é realizada por meio da manuten ção da TABELA DE VARIÁVEIS GLOBAIS pelo CARREGADOR-RELOCÁVEL.

A produção de um MÓDULO DE CARGA no formato RELATIVO (3.4.1.-b), a partir dos MÓDULOS RELOCÁVEIS não difere das opera ções realizadas pelo CARREGADOR-RELOCÁVEL, podendo mesmo este ser adaptado para o fornecimento do arquivo do programa em um ou outro formato.

Uma vez que utilizando-se o formato RELATIVO, o programa quando armazenado irá sofrer uma nova RELACAÇAO (3.4.3), é necessário que o CARREGADOR-RELOCÁVEL assuma uma BASE DE RELOCAÇÃO (BR) igual a O (zero), neste caso.

3.5 - DESCRIÇAO DO PROCESSO DE COMUNICAÇAO (COM)

O processo de comunicação (COM) estabelece uma interação entre o operador, o NÚCLEO e outros processos em andamento no Sistema Básico de Controle.

Esta interação se processa por meio de diretivas e mensagens enviadas através de uma teleimpressora (DEC ou TTY).

As diretivas ou comandos enviados pelo operador são interpretadas e executadas pelo processo COM, após o que este envia uma mensagem ao operador. Na execução dessas diretivas pode haver a intervenção de outros processos, acionados pelo processo COM. Essa interação com outros processos permite ao operador exercer um controle sobre o funcionamento de todo o Sistema Básico de Controle.

3.5.1 - DIRETIVAS EMPREGADAS NO PROCESSO COM

Uma diretiva é constituida por um conjunto de no máximo 40 caracteres ASCII enviados ao processo COM por meio de uma teleimpressora. De acordo com a interpretação e o resultado da execução da diretiva, uma ou mais mensagens são enviadas ao operador através da mesma teleimpressora.

As especificações dos formatos, operações a serem executadas no processo COM a mensagens envolvidas, estão relacionadas na tabela da fig. 3.11.

De um modo geral, cada diretiva é constituida por três caracteres ASCII de designação, seguidos por uma série de parâmetros númericos codificados em notação hexadecimal com 2 digitos, ou 3 digitos precedidos pelo caracter /. Os parâmetros são separados por vírgu

jas. Cada diretiva especifica a execução de um primitivo, correspondente aos caracteres de designação e utilizando os parâmetros relacionados na mesma.

A diretiva EDT,E,/E_1E_0,D_1,D_2,...,D_n faz com que os dados D_1 a D_n sejam convertidos e armazenados consecutivamente a partir do endereço E_1E_0 da memória.

A diretiva EDT,D,/E_1E_0 causa a impressão de 10 posições consecutivas da memória a partir do endereço dado por E_1E_0.

A resposta do processo COM é efetuada através de uma mensagem impressa na teleimpressora de acordo com o resultado da execução da diretiva. Caso esta não possa ser interpretada pelo processo COM, este enviará a mensagem: ?

	CARACTERES DE DESIGNAÇÃO	PARÂMETROS	PRIMITIVO EXECUTADO	MENSAGEM ENVIADA
0	PRT	T	PROGRAMA R.T.T (PRT)	*
1	LRI		LE RELÓGIO INTERNO (LRI)	T_1, T_2, T_3
2	ARI	T_1, T_2, T_3	ALTERA RELÓGIO INTERNO (ARI)	*
3	IES	U	INICIA ENTRADA/SAIDA (IES)	*
4	AES	U	ALTERA ESTADO E/S (AES)	*
5	LES	U	LE ESTADO E/S (LES)	E_1
6	CRP	N	CRIA PROCESSO (CRP)	*
7	AEP	N	ALTERA ESTADO DO PROCESSO (AEP)	*
8	RMP	N	REMOVO PROCESSO (RMP)	*
9	EMS	$N, M_1, M_2 \ldots M_n$	ENVIA MENSAGEM (EMS)	*
10	AMS	N	AGUARDA MENSAGEM (AMS)	$M_1, M_2, \ldots, M_{1\emptyset}$
11	AMG		AGUARDA MENSAGEM (AMG)	$M_1, M_2, \ldots, M_{1\emptyset}$
12	TRP	U	TROCA PRIORIDADE (TRP)	*
13	PXM		PROXIMA MENSAGEM (PXM)	* ou $M_1, M_2, \ldots M_{1\emptyset}$
14	LDP	N	LE DESCRITOR (LDR)	$D_1, D_2, \ldots D_7$
15	EDI	$\underline{E}, /F_1 F_\emptyset, D_1, D_2 \ldots D_X$		*
16	EDT	$\underline{D}, /E, E_\emptyset$		$D_1, D_2, \ldots, D_{1\emptyset}$

Fig. - 3.11

Uma vez executada a diretiva, este envia o caracter * ou uma série de valores relacionados com o primitivo executado. Estas mensagens constituídas por sequências de valores em notação hexadecimal de 2 dígitos e separados por vírgulas estão relacionadas abaixo.

a) T1,T2,T3

Resposta à execução da diretiva LRI.

É impresso o valor da variáveis RIT (Relógio Interno).

b) E

Um código hexadecimal (E) é impresso, correspondente ao estado do dispositivo especificado na diretiva LES,N.

c) M1,M2,...,M10

Este formato corresponde à impressão de uma MENSAGEM enviada ao processo COM, por um outro processo, em resposta à execução de um primitivo ANS, AMG, ou PXM. Estes primitivos são executados de acordo com as diretivas ANS, N ou AMG ou PXM respectivamente.

A execução do processo COM fica suspensa enquanto não receber a MENSAGEM, exceto no caso da diretiva PXM em que o caracter * é impresso se não houver MENSAGEM para o processo COM.

d) D1,D2,...,D7

Impressão dos valores do DESCRITOR de um processo em resposta à diretiva LDP.

3.5.2 - ESTRUTURA DE DADOS DO PROCESSO COM

O processo COM manipula principalmente uma área de dados BES constituída por 40 palavras consecutivas e indicada pelo índice BIN. Nesta área são lidas as diretivas enviadas ao proces

ESTRUTURA DOS DADOS DO PROCESSO COM

BES

BLOCO DE ENTRADA E SAIDA

BIN

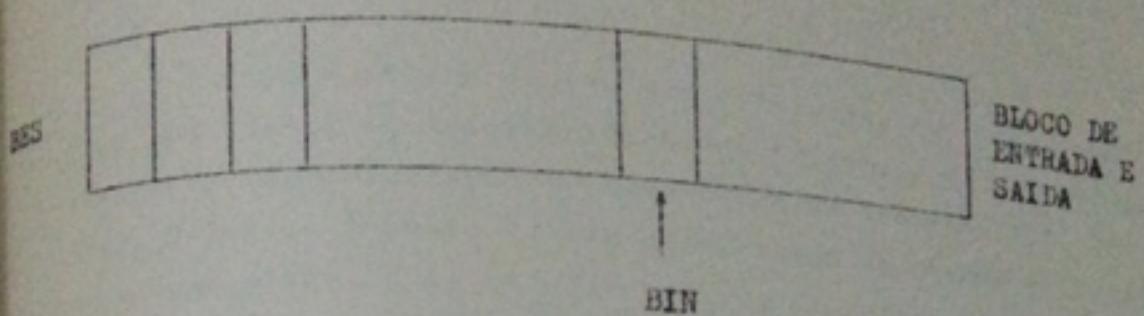



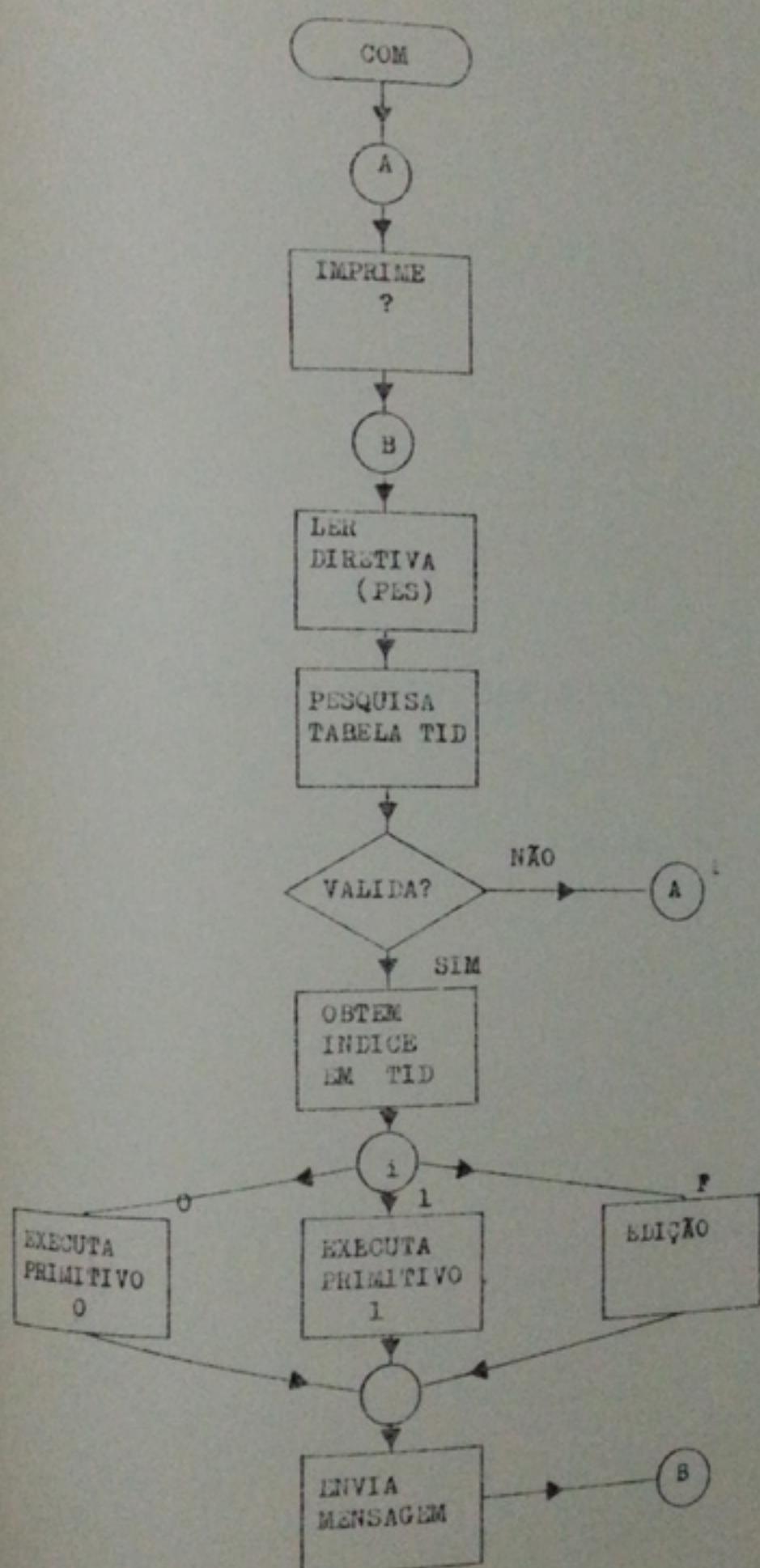
TABELA DE INTERPRETAÇÃO DAS DIRETIVAS (TID)

	TIA	TIB	TIC
0			
i →	D1	D2	D3
P			

Fig. 3.12

so COM, e recebe ainda as mensagens e listagens a serem impres-
sas na teleimpressora (Fig. 3.12).

Para a interpretação das diretivas, o processo COM utiliza uma Tabela de Interpretação das Diretivas (TID) contendo os caracteres de designação de cada diretiva (colunas TIA, TIB e TIC). O índice i nesta tabela corresponde ao número do primitivo a ser executado. A tabela TIM contém, em correspondência a cada diretiva, informações a respeito dos parâmetros a serem tratados.

3.5.3 - ESTRUTURA DO PROCESSO COM

Inicialmente o carácter é impresso assinalando o pedido de leitura de uma diretiva. Por meio do processo PES uma diretiva é lida na área BES e interpretada. Essa interpretação consiste em pesquisar os três caracteres de designação da diretiva na Tabela de Interpretação (TIB). Se a sequência de caracteres fornecida for encontrada nesta tabela, o índice i indicará o primitivo a ser executado, caso contrário, uma nova mensagem é impressa como anteriormente.

De acordo com o índice na tabela TID, uma subrotina especial de execução de cada primitivo em particular é executada.

Após a execução deste primitivo, uma mensagem conveniente é enviada, ainda de acordo com o primitivo que foi executado.

Um diagrama resumo deste procedimento é apresentado na figura 3.13.

Subrotinas de conversão de dados, pesquisa de caracteres e formatação de mensagens são ainda utilizados pelo processo COM.

COM
A
IMPRIME
?
B
LER
DIRETIVA
(PES)
PESQUISA
TABELA TID
VALIDA?
NÃO
A
SIM
OBTEM
INDICE
EM TID
i
0
1
F
EXECUTA
PRIMITIVO
0
EXECUTA
PRIMITIVO
1
EDIÇÃO
ENVIA
MENSAGEM
B

Fig. - 3.13

CAPÍTULO 4 - OBSERVAÇÕES FINAIS

4. - OBSERVAÇÕES FINAIS

Para a definição do Núcleo Básico, descrito no capítulo 2, utilizou-se alguns conceitos já empregados em outros sistemas, sendo que os principais podem ser vistos nas referências (3) e (17). Esses conceitos, tais como os de Primitivos, Processos, Comunicações e Sincronização de Processos, etc, têm caráter geral e podem ser aplicados a qualquer tipo de computador de propósito geral.

Embora a implementação efetuada para o "PATINHO FEIO" tenha utilizado algumas particularidades deste, de um modo geral, esta implementação pode ser realizada em um outro minicomputador com recursos de "Hardware" semelhantes aos do "PATINHO FEIO".

O sistema Básico de Controle (S.B.C.) no entanto, não deve ser encarado como um Sistema Operacional propriamente dito, se considerado o conceito apresentado no capítulo 1, podendo eventualmente servir de ponto de partida para a implementação de algum Sistema Operacional, desde que alguns recursos de "Hardware" e "Software" sejam incorporados.

Os principais recursos adicionais que possam melhorar o desempempenho do Núcleo Básico ou que possam viabilizar a implementação de Sistema Operacional são apontados neste capítulo.

4.1 - MODIFICAÇÕES E AMPLIAÇÕES DO NÚCLEO BÁSICO

O Núcleo Básico está associado diretamente com os recursos de "Hardware" disponíveis. Deste modo, as modificações e ampliações deste Núcleo podem requerer alterações drásticas no B.C.P., hipótese esta que não será considerada aqui.

No entanto, é possível que alterações simples de "Hardware" ou modificações dos programas associados ao Núcleo possam melhorar o desempenho deste.

a) Geração da Interrupção Síncrona

Na execução dos Primitivos, o principal aspecto está associado à geração da interrupção síncrona. O método adotado nesta implementação embora tenha funcionado corretamente, tem o inconveniente de ocupar uma posição de interface das 15 atualmente disponíveis. A inclusão de uma nova instrução que permita a geração dessa interrupção síncrona deverá proporcionar um melhor resultado, liberando uma interface.

b) INCLUSÃO DE NOVOS PRIMITIVOS

O conjunto de Primitivos definidos no Núcleo Básico correspondem a uma primeira avaliação das necessidades de um Sistema que possibilite o controle e intercomunicação de um determinado número de processos. Naturalmente a escolha de um conjunto ótimo de Primitivos irá depender das reais necessidades de um certo grupo de processos que irão implementar um Sistema Operacional e as aplicações deste.

Por esse motivo é possível que alguns dos Primitivos definidos anteriormente devam ser modificados ou que eventualmente outros devam ser incorporados.

Para a inclusão de novos Primitivos é suficiente que as subrotinas que implementam esses Primitivos sejam acrescentados ao conjunto de subrotinas PRN (fig. 2.3) e ainda permitir que estas possam ser referenciadas na rotina de controle ISS.

É importante observar que as estruturas de dados que descrevem e guardam informações relativas ao controle dos processos, estão distribuídas entre a rotina ISS e os códigos relativos aos processos por meio dos DESCRITORES DE PROCESSOS. As informações relativas ao CANAL CONCENTRADOR estão incorporadas às rotinas DCN e as relativas ao N.T.B. à rotina ISS. Qualquer outra estrutura de dados está associada a cada Primitivo em particular, sendo desvinculadas completamente das demais.

Deste modo, um novo Primitivo deve conter uma estrutura de dados própria e não conflitante com aquelas que descrevem os processos, podendo eventualmente fazer uso desta.

c) DESCRIÇÃO DOS PROCESSOS

A descrição dos processos utilizada, supõe a existência de um programa distinto para cada processo. Essa correspondência é estabelecida pelo fato de que o DESCRITOR de um processo acompanha os códigos do programa associados a este.

Este fato não permite que mais de um processo possa ser associado a um único programa, característica esta que pode ser útil em Sistemas de Tempo compartilhado ("Time Sharing") (ref. 5).

Uma modificação nesta descrição que possa permitir esta última característica consiste em colocar os elementos que descrevam o "STATUS" de um processo em colunas adicionais da Tabela de Descrição dos Processos (TDP), mencionada no capítulo 2, em lugar dos apontadores do DESCRITOR. Deste modo, a cada processo fica associado um DESCRITOR independente do programa que o implementa.

Para isto, além da necessária provisão de espaço de memória para as colunas adicionais da tabela TDP, a rotina de Comutação de Processos (CMP) e o primitivo LDP deverão ser modificados convenientemente.

4.1.2 - OPERAÇÕES DE ENTRADA E SAÍDA

A padronização das interfaces ligadas a dispositivos de Entrada/Saída atualmente conectados a "PATINHO FEIO" permitiu que uma única rotina (OCN) pudesse tratar as interrupções provenientes destes dispositivos e realizar as transferências de dados necessárias. Estas interrupções não interferem com o controle da seqüência em que os processos são executados pelo fato de que em nenhuma ocasião estas poderão realizar a Comutação de Processos.

Deste modo, o término da transferência de um conjunto de dados entre a memória e o dispositivo só pode ser verificada quando o processo responsável utilizar os Primitivos de teste associados ao Canal Concentrador (cap. 2).

Este método poderá trazer algum inconveniente se o controle da sequência de processos for mais complexo que o Sistema de "ROUND-ROBIN" adotado.

Uma maneira de se obter uma maior generalidade para o Núcleo consiste em transferir o controle da U.C.P. após o término de uma operação completa de Entrada/Saída, a uma rotina que faça a verificação da necessidade ou não de se realizar uma Comutação de Processos ou então acionar incondicionalmente um processo de Controle de Entrada/Saída (PES).

A inclusão de outros dispositivos se realizada por meio de interfaces do mesmo tipo das utilizadas no CANAL CONCENTRADOR, pode ser realizada sem nenhuma modificação na rotina DCH. Entretanto para dispositivos ligados por meio de interfaces especiais, como por exemplo, a de um Disco Magnético, haverá a necessidade da inclusão de rotinas especiais acionadas por meio da rotina RDL

A inclusão do Disco Magnético é essencial no desenvolvimento de Sistemas Operacionais. Dependendo das características de Entrada/Saída da interface ligada ao Disco, novos primitivos deverão ser incluídos no Núcleo a fim de proporcionar uma maior facilidade na programação das operações associadas a este.

4.1.3 - PROTEÇÃO

Um aspecto importante com relação ao Núcleo Básico, não mencionado até então, refere-se à Proteção de Instruções e à Proteção de Memória.

Uma vez que se têm vários processos simultâneos em anda

mento no Sistema, pode ocorrer que um processo ainda não sufici entemente testado possa comprometer o funcionamento tanto do Nú cleo como dos demais processos, através da execução de instru ções não permitidas como é o caso das instruções de Entrada/Saí da ou por meio de modificações em áreas de memória não pertencen tes ao processo.

Em geral a proteção contra violações deste tipo é pro porcionada pelo "Hardware" do computador em diversas maneiras, co mo por exemplo, a utilizada no minicomputador G-10 (12).

No "PATINHO FEIO" a U.C.P. não dispõe de recursos de proteção deste tipo, havendo necessidade de se incorporar algum mecanismo de proteção que permita um desempenho para o Núcleo bem como de um eventual Sistema Operacional. A proteção de ins truções deverá ser realizada por meio de uma interface especial (18).

4.1.4 - EXPANSÃO DE MEMÓRIA

A possibilidade de Expansão de Memória é o principal fa tor que permite viabilizar uma expansão do Núcleo e a implementa ção de um Sistema Operacional.

O esquema de expansão proposto na referência (18) pode rá ser utilizado inclusive como método de proteção de memória.

A inclusão desse novo recurso deverá inclusive provocar uma reavaliação do Núcleo proposto neste trabalho com a finalida de de adaptá-lo às novas condições impostas por esta expansão.

4.2 - AVALIAÇÃO DO SISTEMA BÁSICO DE CONTROLE

Uma das finalidades da implementação do Sistema Básico de Controle (SBC) definido no capítulo 3 constituiu-se na ilus tração de uma aplicação do Núcleo Básico.

Na implementação do S.B.C. foram utilizados a maioria

dos Primitivos disponíveis, os recursos do Canal Concentrador e ainda foi explorado o fato de que os processos associados ao S.B.C. são executados em partições de tempo ("Time-Slice") em um esquema "ROUND-ROBIN".

Dada a simplicidade deste S.B.C., as ferramentas proporcionadas pelo Núcleo foram mais que suficientes para a sua implementação. Entratanto, para efeito de avaliação de desempenho do Núcleo, as funções proporcionadas pelo S.B.C. não são adequadas para medidas desse desempenho.

Serão necessários a inclusão de alguns processos especiais, por meio do processo COM, que criem situações típicas em sistemas operacionais e realizem as necessárias medidas.

Devido as limitações de memória disponível atualmente, tais processos deverão apenas simular estas situações.

Um exemplo dessa simulação é apresentado no apêndice 2.

APÊNDICE I

APÊNDICE I

Neste apêndice são apresentadas as principais característi-cas das rotinas mais importantes utilizadas na implementação do Núcleo Básico.

Para cada rotina são apresentadas a sua função, variáveis utilizadas, parâmetros de entrada e saída e uma breve descrição de seu funcionamento.

I - ROTINA DE DESCOBRIMENTO DE INTERRUPÇÕES

a) CÓDIGO: RDI

b) FUNÇÃO:

Teste dos pedidos de interrupção das interfaces e implementação do esquema de prioridades.

c) CHAMADA:

Acessível por interrupção

d) PARÂMETROS DE ENTRADA:

Nenhum

e) PARÂMETROS DE SAÍDA:

Nenhum

f) VARIÁVEIS LOCAIS:

Nenhuma

g) VARIÁVEIS GLOBAIS

Nenhuma

h) ROTINAS UTILIZADAS:

ISS, ISR, DCN

i) DESCRIÇÃO:

Realiza uma sequência de testes dos pedidos de interrupção das interfaces e desvios incondicionais para as rotinas responsáveis pelo tratamento de interrupção detectada.

7 - ROTINA DE TRATAMENTO DE INTERRUPÇÕES DO RELÓGIO DE TEMPO REAL

a) CÓDIGO: ISR

b) FUNÇÃO: Tratamento das interrupções geradas pelo Relógio de Tempo Real (R.T.R.)

c) CHAMADA:

Acessível por interrupção do R.T.R.

d) PARÂMETROS DE ENTRADA

Nenhum

e) PARÂMETROS DE SAÍDA

Nenhum

f) VARIÁVEIS LOCAIS

RIT, RIA, RIB - Contador de interrupções

g) VARIÁVEIS GLOBAIS

NDT

h) ROTINAS UTILIZADAS

SST, RST, CMP, PAT

i) DESCRIÇÃO

A cada interrupção do R.T.R. a variável RIT, RIA, RIB é incrementada. Um novo processo é selecionado por meio da rotina PAT e, através da rotina CMP este processo é colocado em execução.

J - ROTINA DE ESCOLHA DO PRÓXIMO PROCESSO ATIVO

a) CÓDIGO: PAT

b) FUNÇÃO:

Busca o próximo processo no estado ATIVO

c) CHAMADA:

PUG PAT

d) PARÂMETROS DE ENTRADA

PEX - Código do processo em execução

e) PARÂMETROS DE SAÍDA

ACUMULADOR: Código do próximo processo ATIVO

f) VARIÁVEIS LOCAIS

Nenhuma

g) VARIÁVEIS GLOBAIS

PEX, TDS

h) ROTINAS UTILIZADAS

PXL

i) DESCRIÇÃO:

A partir do número contido em PEX, a Tabela de Descrição dos processos (TDS) é percorrida. O código do primeiro processo encontrado no estado ATIVO será carregado no ACUMULADOR.

6 - ROTINA DE ATRAZO DE PROCESSO

a) CÓDIGO: ATP

b) FUNÇÃO:

Faz com que um processo repita a emissão de um Primitivo

c) CHAMADA:

PUG ATP

d) PARÂMETROS DE ENTRADA:

Nenhum

e) PARÂMETROS DE SAÍDA:

Nenhum

f) VARIÁVEIS LOCAIS:

Nenhuma

g) VARIÁVEIS GLOBAIS:

NDT

h) ROTINAS UTILIZADAS:

PAT

i) DESCRIÇÃO:

A variável NDT indicadora do número de parâmetros do primitivo é feita igual a -2. Com isto o CONTADOR DE INSTRUÇÕES do processo passará a apontar a instrução correspondente ao último primitivo emitido pelo processo. Um novo processo é acionado por meio de PAT.

5 - ROTINA DE TRATAMENTO DAS INTERRUPÇÕES SÍNCRONAS

a) CÓDIGO: ISS

b) FUNÇÃO:

Tratamento das interrupções síncronas

c) CHAMADA:

Acessível por interrupção

d) PARÂMETROS DE ENTRADA:

Endereço de interrupção (posições 2 e 3 da memória)

e) PARÂMETROS DE SAÍDA:

Nenhum

f) VARIÁVEIS LOCAIS:

EPI, EPJ - conteúdo das posições de interrupção (2 e 3)
NDT - Número de parâmetros transmitidos
NPP - Número de primitivos
TDS - Tabela de Estados dos processos
TDL - Tabela dos Descritores dos processos
TDC - Tabela dos Descritores dos processos
TDN - Tabela de Códigos dos processos

g) VARIÁVEIS GLOBAIS:

Nenhuma

h) ROTINAS UTILIZADAS:

SST, TPI, RST, PRM

i) DESCRIÇÃO:

O "STATUS" do processo que provocou a interrupção é salvo por meio da rotina SST. A variável EPI, EPJ passa a apontar os parâmetros relacionados com o primitivo a ser executado. Por meio da rotina TPI e da variável NDT estes são transferidos. O primei

ro parâmetros é verificado comparando-se com o valor de NPP, após o que uma subrotina de PRM é escolhida de acordo com este parâmetro.

No retorno de ISS, o conteúdo das posições 2 e 3 da memória é acrescido do valor de NDT a fim de atualizar o endereço de retorno da interrupção. O pedido de interrupção da interface síncrona é feito igual a zero, o "STATUS" do processo interrompido é recuperado por meio da rotina RST e finalmente é executada a instrução PPL.

6 - ROTINA DE SALVAMENTO DO "STATUS"

a) CÓDIGO: SST

b) FUNÇÃO:

Salvar provisoriamente o "STATUS" de um processo interrompido.

c) CHAMADA:

PUC SST

d) PARÂMETROS DE ENTRADA:

Nenhum

e) PARÂMETROS DE SAÍDA:

Nenhum

f) VARIÁVEIS LOCAIS:

STA, STE, STI, STT

g) VARIÁVEIS GLOBAIS:

Nenhuma

h) ROTINAS UTILIZADAS:

Nenhuma

i) DESCRIÇÃO:

Os valores do ACUMULADOR, EXTENSÃO e ÍNDICE são armazenados nas variáveis STA, STE e STI respectivamente. Os bits de TRANSBORDO e "VAI-UM" são colocados nos bits 7 e 6 respectivamente da variável STT.

7 - ROTINA DE RESTAURAÇÃO DO "STATUS"

a) CÓDIGO: RST

b) FUNÇÃO:

Restaurar o "STATUS" de um processo interrompido e que deve continuar em execução.

c) CHAMADA:

PUG RST

d) PARÂMETROS DE ENTRADA:

Nenhum

e) PARÂMETROS DE SAÍDA:

Nenhum

f) VARIÁVEIS LOCAIS:

Nenhuma

g) VARIÁVEIS GLOBAIS:

STA, STE, STI, STT

h) ROTINAS UTILIZADAS:

Nenhuma

i) DESCRIÇÃO:

O ACUMULADOR, EXTENSÃO e ÍNDICE são carregados com os valores de STA, STE, STI respectivamente. Os bits de TRANSBORDO e "VAI UM" são atualizados conforme a variável STT.

8 - ROTINA DE TRANSFERÊNCIA DE PARÂMETROS DE INTERRUPÇÃO

a) CÓDIGO: TPI

b) FUNÇÃO:

Transferência dos parâmetros associados com a execução de um primitivo.

c) CHAMADA:

PUG TPI

d) PARÂMETROS DE ENTRADA:

Nenhum

e) PARÂMETROS DE SAÍDA:

ACUMULADOR

f) VARIÁVEIS LOCAIS:

Nenhuma

g) VARIÁVEIS GLOBAIS:

EPI, EPJ, NDT

h) ROTINAS UTILIZADAS:

Nenhuma

i) DESCRIÇÃO:

A Variável NDT é utilizada como um índice para a sequência de parâmetros de um primitivo, sequência esta designada pelo endereço contido em EPI, EPJ.

A cada chamada de TPI, o ACUMULADOR é carregado com o parâmetro indicado por NDT utilizando-se o endereçamento indireto pós-indexado. Finalmente o valor de NDT é incrementado.

9 - ROTINA DE ARMAZENAMENTO DE PARÂMETROS DE INTERRUPÇÃO

a) CÓDIGO: API

b) FUNÇÃO:

Armazenar parâmetros na sequência de chamada de um primitivo após a execução deste.

c) CHAMADA:

PUG API

d) PARÂMETROS DE ENTRADA:

ACUMULADOR

e) PARÂMETROS DE SAÍDA:

Nenhum

f) VARIÁVEIS LOCAIS:

Nenhuma

g) VARIÁVEIS GLOBAIS:

EPI, EPJ, NDT

h) ROTINAS UTILIZADAS:

Nenhuma

i) DESCRIÇÃO:

A sequência de parâmetros de um primitivo é apontada pela variável EPI, EPJ, NDT indica a posição do parâmetro nesta sequência. Utilizando o endereçamento indireto pós-indexado, o valor do ACUMULADOR é armazenado nesta posição indicada por EPI,EPJ e NDT. A variável NDT é incrementada.

10 - ROTINA DE BUSCA NA LISTA DOS PROCESSOS

a) CÓDIGO: PXL

b) Função:

Percorrer a lista de processos TDS.

c) CHAMADA:

PUG PXL

d) PARÂMETROS DE ENTRADA:

ACUMULADOR: identificação do processo a partir do qual a lista deve ser percorrida.

e) PARÂMETROS DE SAÍDA

ACUMULADOR: identificação do próximo processo na lista TDS.

ÍNDICE: identificação do processo de partida

f) VARIÁVEIS LOCAIS:

Nenhuma

g) VARIÁVEIS GLOBAIS:

TDS

h) ROTINAS UTILIZADAS:

Nenhuma

i) DESCRIÇÃO:

O valor contido em TDS na posição indicada pelo ACUMULADOR é separado e a parte correspondente ao encadeamento da lista (4 bits menos significativos) é devolvida no ACUMULADOR.

a) CÓDIGO: PQN

b) FUNÇÃO:

Pesquisar na tabela de Descrição dos Processos (TDP) a identificação de um processo dado a seu código.

c) CHAMADA:

PUG PQN

d) PARÂMETROS DE ENTRADA:

ACUMULADOR: código do processo

e) PARÂMETROS DE SAÍDA:

ACUMULADOR: Identificação do processo. Se o processo não existir o ACUMULADOR é devolvido com o valor zero.

f) VARIÁVEIS LOCAIS:

Nenhuma

g) VARIÁVEIS GLOBAIS:

TDS, TDN

h) ROTINAS UTILIZADAS:

PXL

i) DESCRIÇÃO:

Os códigos de cada processo, contidos na tabela TDN são comparados com código fornecido. A pesquisa é realizada utilizando-se a rotina PXL. Quando o código for encontrado, a identificação do processo é automaticamente fornecida por PXL.

12 - ROTINA DE MANIPULAÇÃO DE LISTAS

a) CÓDIGO: PQL

b) FUNÇÃO:

Retirar e inserir elementos em uma lista encadeada

c) CHAMADA:

PUG PQL

d) PARÂMETROS DE ENTRADA:

ACUMULADOR :

F=0 INSERE
F=1 RETIRA
L=0 LISTA DE PROCESSOS (TDS)
L=1 LISTA DE MENSAGENS (TML)
L=2 LISTA DE PROCESSOS SUBORDINADOS (TDH)

ÍNDICE: posição a ser retirada da lista

e) PARÂMETROS DE SAÍDA:

ACUMULADOR:

ACUMULADOR > 0 - posição na lista onde um elemento foi acrescentado

ACUMULADOR=0 - ERRO

f) VARIÁVEIS LOCAIS:

PQI - Variável auxiliar
PQT - endereços das listas manipuladas
PQQ - endereço da lista sendo manipulada
ILE - posição do primeiro elemento da lista
ILV - posição do primeiro elemento da parte vazia da lista

g) VARIÁVEIS GLOBAIS:

TDS, TML, TDH

h) ROTINAS UTILIZADAS:

Nenhuma

i) DESCRIÇÃO:

As listas TDS, TML e TDH são estruturadas de forma idêntica.

Cada uma é constituída por uma tabela contendo 16 palavras. Uma posição na lista corresponde a um índice nesta tabela. A posição de índice zero é reservada para o controle da tabela enquanto que cada uma das 15 restantes são subdivididas em duas partes. Os quatro bits mais significativos conterão uma informação particular da lista e os quatro bits menos significativos são reservados para o encadeamento da lista.

Cada tabela comporta duas listas, a Lista de Elementos (LE) e a Lista Vazia (LV).

A lista de elementos inicia-se em uma posição da tabela indicada pelos quatro bits mais significativos da posição zero, enquanto que os quatro bits menos significativos irão indicar a posição inicial da Lista Vazia. O encadeamento consiste em dada uma posição na lista, a posição do próximo elemento desta lista será obtido por meio dos quatro bits menos significativos. Em particular, se o valor desses quatro bits forem nulos, indicarão o término da lista.

Um esquema desta tabela é indicado na figura 2.6.

A rotina PQL inicialmente armazena o ÍNDICE na variável PQI e por meio de uma análise do conteúdo do ACUMULADOR atualiza a variável PQT que irá indicar a tabela a ser processada (TDS, TML ou TDB) e ainda distingue a operação a ser realizada. Obtido o endereço da tabela, as variáveis ILE e ILV serão atualizadas com a posição do primeiro elemento da Lista de Elementos e da Lista Vazia respectivamente.

Realizada a iniciação dessas variáveis, as operações serão realizadas em secções diferentes da rotina.

No caso de inserção de um novo elemento na Lista de Elementos, este será colocado na primeira posição da Lista Vazia, isto é, no retorno da rotina, o ACUMULADOR é carregado com ILV e o encadeamento das duas listas é atualizado.

No caso de retirada de um elemento, a Lista de Elementos é percorrida até que seja encontrada a posição indicada em PQI. O encadeamento é atualizado de forma que essa posição passe a pertencer à Lista Vazia.

A condição de erro somente ocorre quando for pedida a inserção de um novo elemento e a Lista de Elementos estiver ocupando todas as posições da tabela, caso em que ILV=0.

As inserções e retiradas de elementos destas tabelas referem-se apenas à manipulação das listas enquanto que a atualização de informações associadas à tabela são realizadas externamente à rotina conhecendo-se as posições dos elementos na tabela.

a) CÓDIGO: CMP

b) FUNÇÃO:

comutação de processos

c) CHAMADA:

PUG CMP

d) PARÂMETROS DE ENTRADA:

ACUMULADOR: identificação do próximo processo

e) PARÂMETROS DE SAÍDA:

Nenhum

f) VARIÁVEIS LOCAIS:

PEX: identificação do processo em execução
PDI, PDJ: endereço do Descritor do processo

g) VARIÁVEIS GLOBAIS:

TDL, TDC, NDT, STA, STE, STI, STT

h) ROTINAS UTILIZADAS:

Nenhuma

i) DESCRIÇÃO:

Por meio da variável PEX é obtida a identificação do processo em execução. Com essa identificação o endereço do Descritor deste processo é lançado nas variáveis PDI e PDJ consultando-se as tabelas TDL e TDC. Os valores do ACUMULADOR, EXTENSÃO, ÍNDICE, CONTADOR DE INSTRUÇÕES, TRANSBORDO (T) e "VAI-UM" (V) são lidos das variáveis STA, STE, STI, posições 2 e 3 da memória e STT respectivamente e lançados no Descritor por meio do endereço

PDI, PDJ. Essas variáveis foram atualizadas por ocasião da interrupção do processo indicado por PEX.

Em seguida, a variável PEX é feita igual ao valor do ACUMULADOR transferido para a rotina e da mesma forma anterior o endereço do novo DESCRITOR é colocado nas variáveis PDI e PDJ. Os valores contidos neste DESCRITOR são transferidos para as variáveis STA, STE, STI, STT e para as posições 2 e 3 da memória.

A comutação somente será efetivada acionando-se a rotina de Restauração de "STATUS" (RST) e pela execução da instrução PPL. Estas duas últimas operações são realizadas externamente à rotina CMP.

A rotina CMP não é utilizada quando um processo for interrompido e acionado novamente em seguida.

14 - ROTINA DE MUDANÇA DE ESTADO DE UM PROCESSO

a) CÓDIGO: RTE

b) FUNÇÃO:

Modificar o Estado de um processo

c) CHAMADA:

PUG RTE

d) PARÂMETROS DE ENTRADA:

ACUMULADOR: Estado a ser armazenado
ÍNDICE: Identificação do processo

e) PARÂMETROS DE SAÍDA:

Nenhum

f) VARIÁVEIS LOCAIS:

Nenhuma

g) VARIÁVEIS GLOBAIS:

TDS

h) ROTINAS UTILIZADAS:

Nenhuma

i) DESCRIÇÃO:

Por meio do valor do ÍNDICE fornecido, a correspondente posição na tabela TDS é atualizada de acordo com o valor transmitido no ACUMULADOR.

a) CÓDIGO: PRM

b) FUNÇÃO:

Execução dos primitivos

c) CHAMADA:

PUG PRT ou
PUG LRI
PUG ARI
PUG IES
PUG AES
PUG LES
PUG CRP
PUG AEP
PUG EMS
PUG AMS
PUG AMG
PUG PXM
PUG RMP
PUG LDP
PUG TRP

d) PARÂMETROS DE ENTRADA:

Nenhum

e) PARÂMETROS DE SAÍDA:

Nenhum

f) VARIÁVEIS LOCAIS:

Definidas para cada primitivo

g) VARIÁVEIS GLOBAIS:

RDI, PEX, TDS, TDL, TDC, TDN, TEE, TEB, TEC

h) ROTINAS UTILIZADAS:

PQL, PQN, ATP, CMP, RTE, TPI, API

i) DESCRIÇÃO:

Cada primitivo é processado por uma subrotina pertencen te a PRM. Os parâmetros associados a cada primitivo são transferi dos por meio das rotinas TP1 e API. Esses parâmetros estão coloca dos em uma sequência que acompanha a instrução de geração da in terrupção síncrona e serão designados pelos rótulos P_1, P_2,... P_n, e conterão os valores x_1, x_2, ... x_n.

```
        CSP
        DEFC    N
P1      DEFC    X1
P2      DEFC    X2
P3      DEFC    X3
 .               .
 .               .
 .               .
Pn      DEFC    Xn
```

SUBROTINA PRT

X_1 é enviado à interface do RTR por meio de uma instru ção "SAI".

SUBROTINA LRI

As variáveis RIT, RIA e RIB são colocadas nas posições P_1, P_2 e P_3 respectivamente.

SUBROTINA ARI

Os valores X_1, X_2 e X_3 são colocados nas variáveis RIT, RIA e RIB respectivamente.

SUBROTINA AES

Uma linha de índice X_1 das tabelas TEE, TEB e TEC ("CA NAL CONCENTRADOR") são atualizadas com os valores X_2, X_3, X_4 res pectivamente.

SUBROTINA IES

A interface designada por X_1 é acionada por meio de instruções de Entrada/Saída conforme a operação indicada na tabela TEE.

SUBROTINA LES

O conteúdo da linha especificada por X_1, da tabela TEE é transferido para P_2.

SUBROTINA TRP

As instruções de testes de pedidos de interrupção realizadas pela rotina RDI e ainda os endereços de desvio que acompanham estes testes são trocados de acordo com as posições indicadas pelo parâmetro X_1.

SUBROTINA CRP

Por meio da rotina PQN, o código do processo a ser criado (X_1) é verificado. Caso já exista um processo com esse código, a rotina ATP é acionada.

Através da rotina PQL é pedida a inclusão de um novo elemento na tabela TDS. Obtida a posição do novo elemento, as tabelas TDN, TDL e TDC são atualizadas nessa posição com os valores de P_1, P_2 e P_3 respectivamente. A posição da tabela TDS correspondente ao Estado do novo processo é preenchida com $(20)_{16}$ correspondendo ao Estado PARADO.

Se inicialmente a tabela TDS já estiver totalmente preenchida a rotina ATP será acionada.

SUBROTINA AEP

A posição na tabela TDS do processo designado por X_1 é obtida através da rotina PQN. Os quatro bits mais significativos dessa posição de TDS são preenchidos com o valor de X_2.

SUBROTINA LDP

Os valores das tabelas TDL e TDC na posição indicada

pelo código do processo (X_1) são transferidos para P_2 e P_3 respectivamente. A posição nas tabelas é obtida por meio da rotina PQN.

SUBROTINA RMP

Como nas demais subrotinas, a posição nas tabelas de Descrição dos Processos é obtida por meio da rotina PQN conforme o parâmetro X_1. Representando-se por i essa posição, os conteúdos dessas tabelas serão designados por TDS (i), TDL (i), TDC (i) e TDN (i).

Para a remoção dos processos é utilizada a tabela TDN, aonde serão assinalados os processos que devem ser removidos, uma vez que a remoção de um processo i acarreta a remoção de seus subordinados.

A remoção de cada processo é realizada acionando-se a rotina PQL, com o fornecimento da posição do processo na tabela TDS.

A pesquisa de todos os processos a serem removidos é realizada por meio da tabela de processos subordinados TDH, uma vez que esta tabela contém, para cada processo, uma lista encadeada com a indicação de seus subordinados.

Por meio dos quatro bits mais significativos de TDI (i) a lista TDH é percorrida e para cada processo j desta lista é feito TDN (j)=0.

Finalmente, a lista TDS é percorrida e para todo processo no qual TDN (k)=0 é acionado a rotina PQL com o parâmetro k.

SUBROTINA EMS

As subrotinas que processam as MENSAGENS trocadas pelos processos utilizam as tabelas TMD, TML e TMJ descritas no capítulo 2.

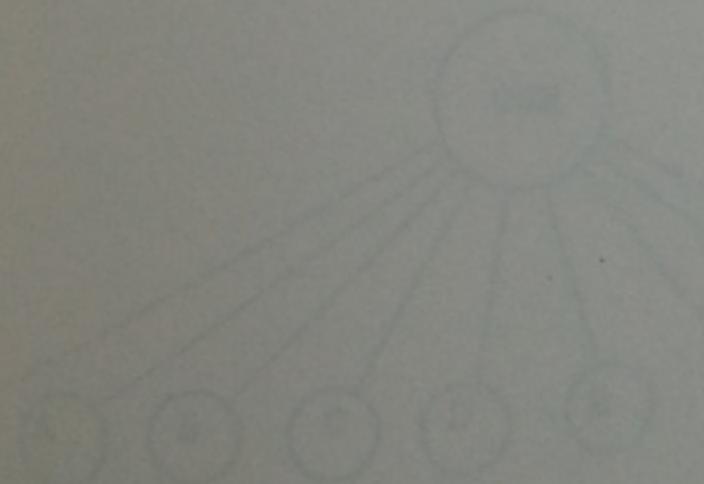
O processo REMETENTE, obtido através de PEX, e DESTINATÁRIO por meio de X_1 e rotina PQN.

Em seguida, uma posição na tabela TML é obtida através da rotina PQL e TDM (i) é feito igual a DR enquanto que TML (i) e TMJ (i) receberão os valores X_2 e X_3 respectivamente.

Caso não exista posição vaga na tabela TML, a rotina ATP é acionada.

SUBROTINA AMS

Uma variável DR é obtida através da variável PEX e do parâmetro X_1 (rotina PQR).

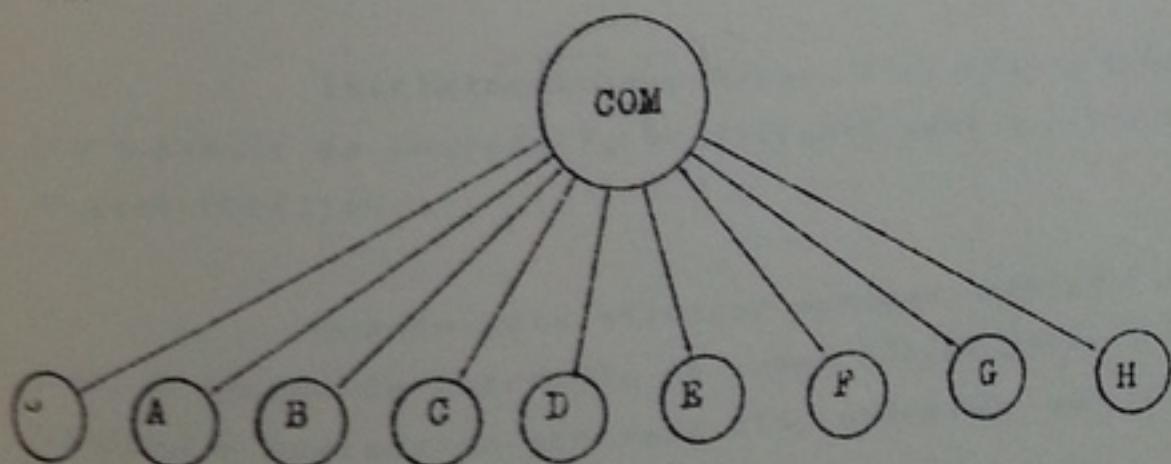
O valor de DR é pesquisado na tabela TMD. Se for encontrado, as posições correspondentes das tabelas TML e TMJ são armazenadas em P_2 e P_3. Caso contrário é acionada a rotina ATP e RTE, a fim de colocar o processo indicado por PEX no estado de ESPERA.

SUBROTINA AMG

Utiliza a mesma variável DR da subrotina AMS, exceto que a parte correspondente à identificação do processo REMETENTE é feita igual a zero. A tabela TMD é pesquisada somente quanto à identificação do processo DESTINATÁRIO. O restante da subrotina é idêntica à subrotina AMS.

SUBROTINA PXM

É idêntica à subrotina AMG, exceto no caso em que DR não é encontrado na tabela TMD. Se isso ocorrer os parâmetros P_2 e P_3 são feitos iguais a -1.

APÊNDICE 2

1. - EXEMPLO PARA SIMULAÇÃO

Um conjunto de 8 processos devem utilizar um recurso comum disponível no sistema. Este recurso, representado por uma impressora de linhas, é atribuído sequencialmente a cada um dos processos mediante uma requisição enviada a um outro processo su pervisor.

Aos processos usuários, denominados por A, B, C, D, E, F, G, H, são atribuídas prioridades com relação à utilização da impressora, de tal modo que o processo supervisor S, recebendo uma requisição, deve colocar o processo requisitante em uma "fila" de acordo com a prioridade do mesmo. A atribuição da U.C.P. a cada um dos processos é suposta realizada em um esquema "ROUND ROBIN", não existindo prioridades em relação à utilização da U.C.P.

2. - O SIMULADOR

Um possível sistema que possa realizar as tarefas pro postas anteriormente foi organizado de acordo com o esquema abai- xo:

2.1 - O PROCESSO S

O processo S mantém uma "fila", realizada por meio de uma tabela TPR constituída por 8 posições.

Cada posição de TPR está associada a um dos processos, de tal modo que a posição de índice 7 corresponde ao processo H, sendo esta posição a de maior prioridade.

O conteúdo de cada posição de TPR indica se o processo correspondente deseja ou não utilizar a impressora. No primeiro caso TPR(I)≠0 e no segundo caso TPR(I)=0, sendo mantida a seguinte correspondência:

POSIÇÃO I	PROCESSO
0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H

Inicialmente o processo S examina a tabela TPR, sempre a partir da posição 7, verificando qual a primeira posição I na qual TPR(I)≠0.

Quando esta situação ocorrer, TPR(I) é feita igual a zero, o número do intervalo de tempo ("time slice") é lido e uma MENSAGEM é enviada ao processo PES, ordenando que seja impressa uma linha.

Esta linha conterá o nome do processo, o valor do contador correspondente à posição I na tabela CNT e o número do intervalo de tempo lido anteriormente.

Em seguida, uma MENSAGEM é enviada ao processo correspondente a I.

Novamente, a tabela TPR é consultada e um novo ciclo é realizado, conforme indicado na figura A2-1.

2.2. - OS PROCESSOS A, B, C, D, E, F, G e H

Uma característica desses processos é que eles compartilham um mesmo programa. Este programa, denominado por P, não altera o valor do registrador ÍNDICE (I) de modo que quando em execução o valor deste índice I indicará qual dos processos que o utiliza.

Assim, os DESCRITORES dos processos A até H conterão valores fixos para o ÍNDICE e indicarão através do CONTADOR DE INSTRUÇÕES (CI) uma mesma área de códigos.

O comportamento do programa P refletirá portanto o comportamento de qualquer um dos processos A até H.

A área de dados associada a este programa é constituída por três tabelas TPR, CNT e PX contendo 8 posições cada uma. Essas tabelas são utilizadas sempre por meio de instruções " indexadas " de tal forma que os dados TPR(I), CNT(I) e PX(I) são reservados ao processo com índice I. Desse modo, o processo A, por exemplo, utiliza somente os valores TPR(0), CNT(0) e PX(0).

Cada um desses processos deverá somar 1 no contador CNT(I) e comparar o resultado com o limite PX(I). Se CNT(I) for menor que PX(I) uma requisição da impressora é enviada ao processo,

S, fazendo TPR(I) diferente de zero. Após esta requisição, uma MENSAGEM do processo S é aguardada. Recebida a MENSAGEM de S o ciclo é repetido. Quando o contador exceder o valor limite (PXC1) o processo passará a aguardar uma MENSAGEM do processo de comunicação COM.

Se receber uma MENSAGEM de COM, esta deverá conter um novo valor limite de contagem que será armazenado em PX(I). Neste caso o contador é reiniciado com zero e o processo retoma a contagem descrita anteriormente. A figura A2.2 esquematiza esses procedimentos.

3. - SIMULAÇÃO

Os programas associados aos processos S, A, B, C, D, E, F, G, e H denominados SSS e P foram escritos na linguagem do "MONTADOR RELOCÁVEL" e processados pelo "CARREGADOR RELOCÁVEL" descritos na referência (2) e (15) obtendo-se uma fita perfurada em "FORMATO ABSOLUTO" com início na posição de endereço $(F00)_{16}$ da memória. Os Descritores dos processos A até H, ficaram localizados nas posições dadas pela tabela abaixo, juntamente com os códigos associados a esses processos.

PROCESSO	ENDEREÇO DO DESCRITOR	CÓDIGO
S	$(E00)_{16}$	5
A	$(EE6)_{16}$	41
B	$(EED)_{16}$	42
C	$(EF4)_{16}$	43
D	$(EF8)_{16}$	44
E	$(E02)_{16}$	45
F	$(E09)_{16}$	46
G	$(F10)_{16}$	47
H	$(F17)_{16}$	48

A utilização do Sistema Básico de Controle para esta simulação é demonstrada no final deste apêndice, onde comparecem os comandos enviados ao SBC ao lado de alguns comentários e o resultado obtido.

Esses comandos podem ser agrupados em:

a) IMPRESSÃO DO TÍTULO

Um cabeçalho constituído por:

PROCESSO CONTADOR * INSTANTE

foi inicialmente perfurado em uma fita de papel e colocado na leitora de fitas. Por meio do envio de uma MENSAGEM ao processo PES(2) na forma

EMS,2,4,2,0F,20,50

o conjunto de caracteres do Título foi armazenado em posições a partir do endereço (F20) da memória.

Alguns caracteres de controle da impressora foram acrescentados pelo comando EDT e novamente através de uma MENSAGEM enviada ao processo PES,

EMS,2,4,64,0F,20,26

obteve-se a impressão deste Título na impressora de linhas.

b) CARGA DOS PROGRAMAS SSS e P

A fita perfurada contendo os programas S9I e P foi transferida para a memória pelo processo CPR(3) através do comando de envio de Mensagens.

EMS,3,4,0,2,0E,00

Para a verificação do transporte destes programas, o processo COM foi colocado no estado de "Espera" aguardando uma MEN

SAGEM de CPR, por meio do comando

AMS,3

Recebida a Mensagem de CPR, esta foi impressa verifi-
cando-se que a transferência foi correta uma vez que a primeira pa
lavra da Mensagem é zero.

c) CRIAÇÃO DOS PROCESSOS

Conhecendo-se as posições dos Descritores dos proces
sos S e A até H, estes forma adicionados ao SBC por meio do coman
do CRP.

d) ATIVAÇÃO DOS PROCESSOS

Uma vez que os processos recém criados permanecem no
estado "PARADO", estes devem ser ativados mediante um comando de
"ALTERA ESTADO" AEP, onde se especifica o código do processo e o
novo estado assumido.

O processo S foi mantido no estado "PARADO".

e) INICIAÇÃO DOS PROCESSOS A,B,C,D,E,F e H

As tabelas CNT e PX relativas ao programa P foram ini
ciadas de tal forma que na primeira vez que esses processos as con
sultam, passem a aguardar uma mensagem do processo COM que irá con
ter o número de vezes em que a impressão deverá ser requisitada ao
processo S.

Desta forma a iniciação desses processos corresponde
ao envio de Mensagens, pelo processo COM, contendo este número.

Os comandos EMS foram utilizados especificando o código do processo e o número escolhido.

f) ATIVAÇÃO DO PROCESSO S

Após a iniciação dos processos A até H, passaram a requisitar a impressora. Entretanto nada foi impresso até que o processo S fosse ativado.

O número do intervalo de tempo foi iniciado com zero por meio do comando ARI,0,0,0 e em seguida o processo S foi ativado pelo comando AEP,5,80 iniciando a utilização da impressora.

ESTRUTURA DO PROCESSO S

Fig. - A2.1

ESTRUTURA DO PROGRAMA P

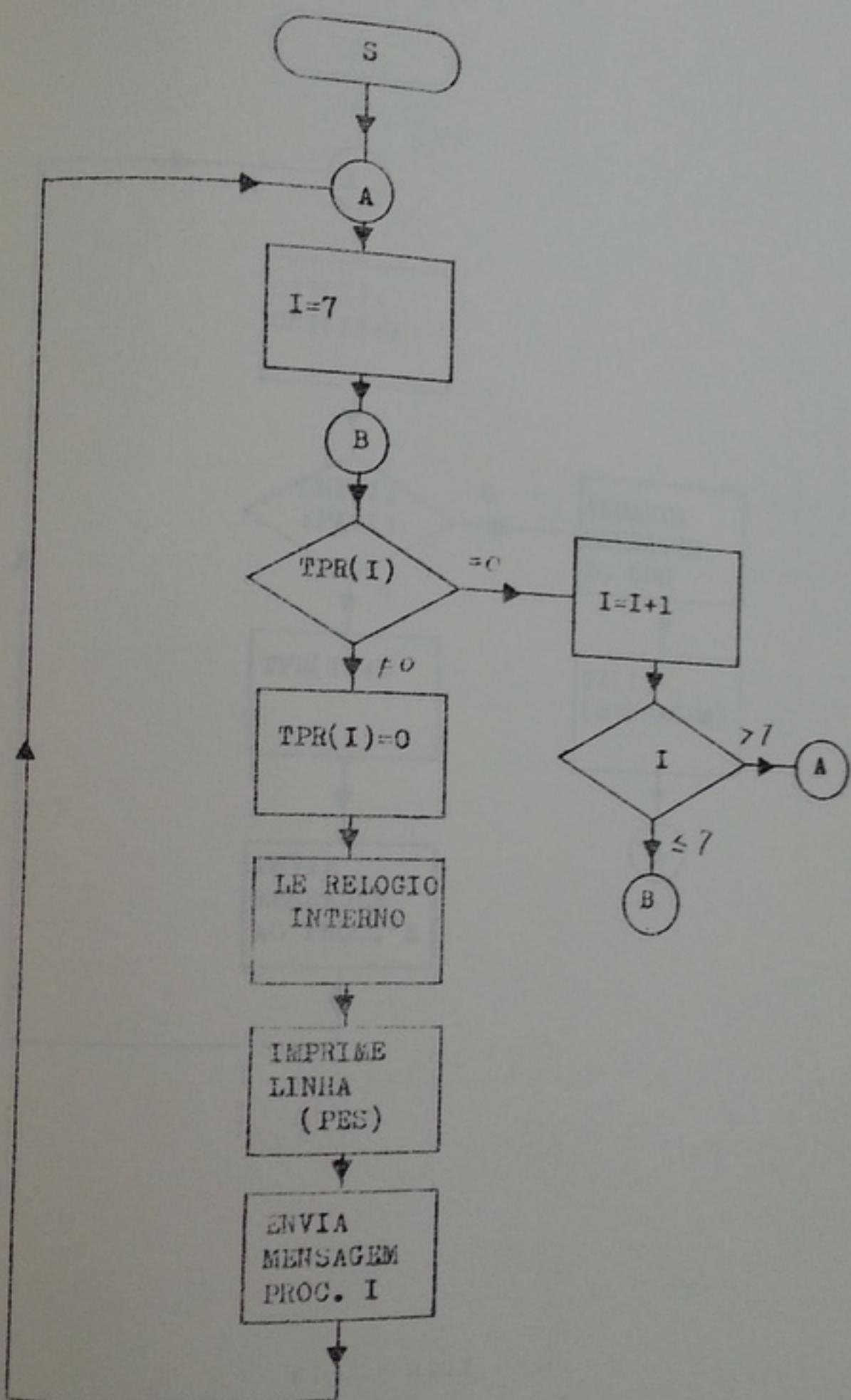
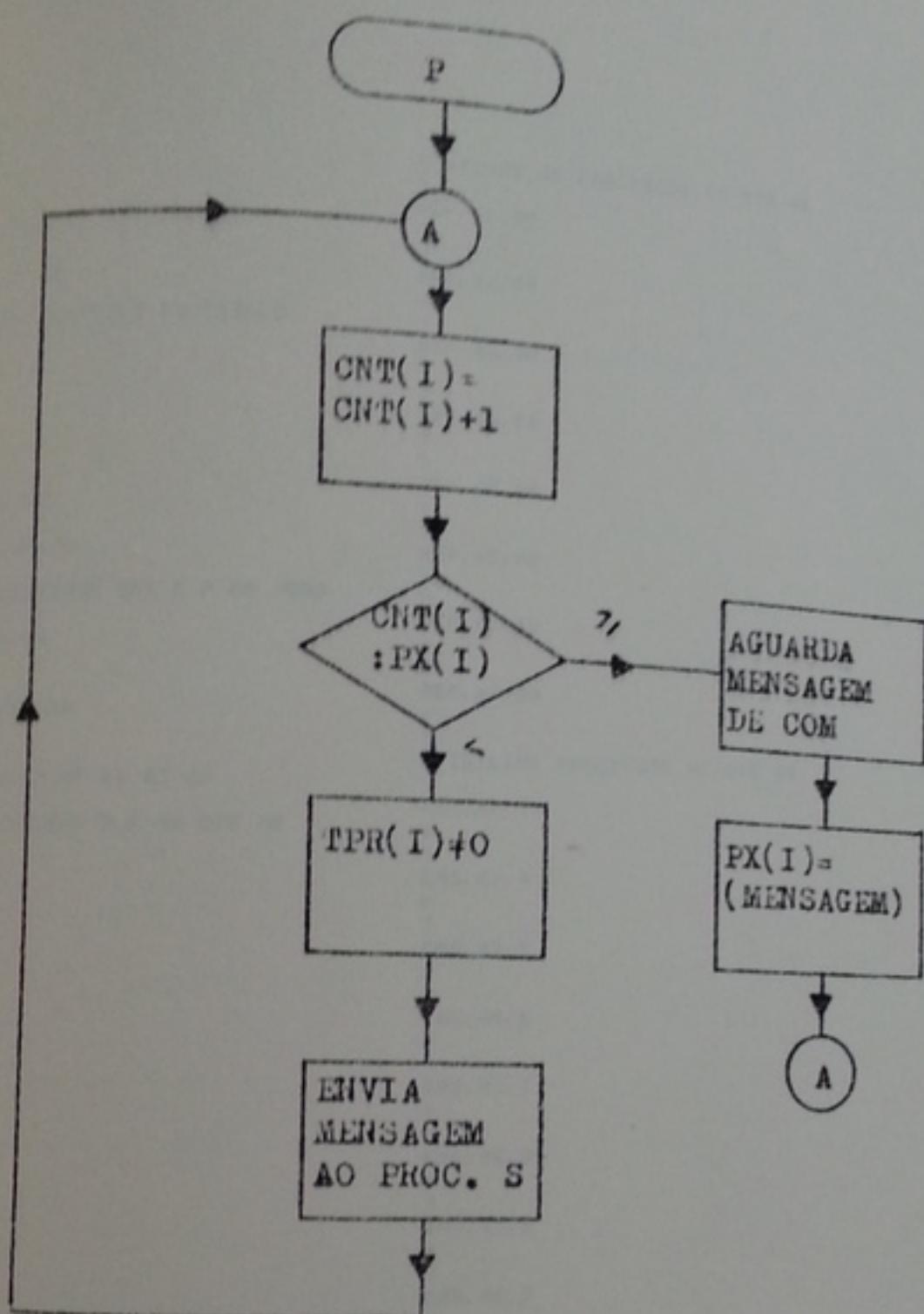



Fig. - A2.2

```
?
C:LEITURA DO TITULO
?
EMS,2,4,2,0F,20,50
*
C:CARACTERES DE CONTROLE DO TITULO
?
EDT,E,/F20,84
*
?
EDT,E,/F45,82
*
?
C:IMPRIMIR TITULO
?
EMS,2,4,64,0F,20,26
*
C:CARGA DOS PROGRAMAS SSS E P EM /E00
?
EMS,3,4,0,2,0E,00
*
?
C:VERIFICAR A CARGA
?
AMS,3
00 00 00 00 02 00 0F 80 03 22
?
C:CRIAR OS PROCESSOS 5 E 41 ATE 48
?
CRP,5,/E00
*
?
CRP,41,/EE6
*
?
CRP,42,/EED
*
?
CRP,43,/EF4
*
?
CRP,44,/EFB
*
?
CRP,45,/F02
*
?
CRP,46,/F09
*
?
CRP,47,/F10
*
?
CRP,48,/F17
*
?
```

```
C:ATIVAR OS PROCESSOS 41 ATE 48
?
AEP,41,80
*
?
AEP,42,80
*
?
AEP,43,80
*
?
AEP,44,80
*
?
AEP,45,80
*
?
AEP,46,80
*
?
AEP,47,80
*
?
AEP,48,80
*
?
C:INICIAR PROCESSOS 41 ATE 48
?
EMS,41,3
*
?
EMS,42,4
*
?
EMS,43,5
*
?
EMS,44,6
*
?
EMS,45,7
*
?
EMS,46,8
*
?
EMS,47,9
*
?
EMS,48,9
*
?
C:INICIAR RELOGIO INTERNO
?
ARI,0,0,0
*
?
C:ATIVAR PROCESSO 5
?
AEP,5,80
*
?
```

```
PROCESSO  CONTADOR * INS

 H  01 = 00004C
 G  01 = 00004E
 H  02 = 000052
 G  02 = 000054
 H  03 = 000058
 G  03 = 00005A
 H  04 = 00005E
 G  04 = 000060
 H  05 = 000064
 G  05 = 000066
 H  06 = 00006A
 G  06 = 000064
 H  07 = 000070
 G  07 = 000072
 H  08 = 000076
 G  08 = 000078
 F  01 = 00007C
 E  01 = 00007E
 F  02 = 000082
 E  02 = 000084
 F  03 = 000088
 E  03 = 00008C
 F  04 = 00008E
 E  04 = 000092
 F  05 = 000094
 E  05 = 000098
 F  06 = 00009A
 E  06 = 00009E
 F  07 = 0000A0
 D  01 = 0000A4
 C  01 = 0000A6
 D  02 = 0000AA
 C  02 = 0000AC
 D  03 = 0000B0
 C  03 = 0000B2
 D  04 = 0000B6
 C  04 = 0000B8
 D  05 = 0000BC
 B  01 = 0000C0
 A  01 = 0000C2
 B  02 = 0000C6
 A  02 = 0000C8
 B  03 = 0000CC
```

R E F E R Ê N C I A S

(1) FREGNI, EDSON
"Projeto Lógico da Unidade de Controle de um Minicomputador"
Dissertação de mestrado - EPUSP, 1972

(2) NETO, JOÃO JOSÉ
"Aspectos do Projeto de Software de um Minicomputador"
Dissertação de Mestrado - EPUSP, 1975

(3) HANSEN, BRINCH
"Operating Systems Principles"
Prentice-Hall, INC. - 1973

(4) FARWELL, RICHARD
"Operating Systems: The Key to Minicomputer Systems"
Data General Corporation - 1973

(5) DENNING, PETER J.
"Third Generation Computer Systems"
Computing Surveys, vol. 3, nº 4, December - 1971

(6) HEWLETT-PACKARD
"Real Time Software"
September - 1969

(7) MILLS, DAVID L.
"Executive Systems and Software Development for Minicomputers"
Proceedings of the IEEE, vol. 61, nº 11, November - 1973

(8) LANGDON JR., GLEN GEORGE e FREGNI, EDSON
"Projeto de Computadores Digitais"
Edgard Blücher, Editora da U.S.P. - 1974

(9) TACHIBANA, MARIO
"Carregador Relocável para o Computador PATO FEIO" - 1974.
Publicação Interna do Lab. de Sistemas Digitais

(10) DONOVAN, JOHN J. e MADNICK S.
"Operating Systems"
McGraw-Hill - 1975

(11) KOVACH, STEPHAN
"Projeto de um Sistema de Entrada e Saída para um Minicomputador"
Dissertação de Mestrado - 1975.

(12) MINICOMPUTADOR G-10
"Manual de Arquitetura - G-10-M01"
Digibrás - agosto 1975

(13) HEWLETT-PACKARD
"Basic Control System"
February - 1968

(14) SCHOEFFLER, JAMES D. e BRONNER LEE R.
"Data Management Software for Mini Production Monitoring and Control Systems"
Proceedings of the IEEE, vol. 61, nº 11, november - 1973

(15) MINICOMPUTADOR G-10
"Manual de Operação G-10-M05"
Digibrás - agosto 1975

(16) MINICOMPUTADOR G-10
"Manual do Montador - G-10-M04"
Digibrás - agosto 1975

(17) LIN, CHARLIE
"Carregador Relocável para o PATINHO FEIO: Versão em 1 Passo"
Publicação interna do Lab.de Sistemas Digitais - 1976

(18) MARTUCCI JR, MOACYR
"Projeto de uma Interface de Controle para Memória Monolitica: Recurso para Ampliação da Memória Principal um Minicomputador".
Dissertação de Mestrado - a ser publicada.